Statistical Image Sequence Segmentation Using Multidimensional Attributes

by

EDMOND CHALOM

B.S., Massachusetts Institute of Technology (1988)
M.S., Massachusetts Institute of Technology (1989)
E.E., Massachusetts Institute of Technology (1992)

Submitted to the Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the MASSACHUSETTS INSTITUTE OF TECHNOLOGY
January 1998

Author .....
Edmond Chalom
Department of Electrical Engineering and Computer Science
January 30, 1998

Certified by .....
V. Michael Bove, Jr.
Principal Research Scientist, MIT Media Laboratory
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Departmental Committee on Graduate Students

Statistical Image Sequence Segmentation Using Multidimensional Attributes

by

Edmond Chalom

Submitted to the Department of Electrical Engineering and Computer Science
on January 30, 1998, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

A novel interactive approach is presented for finding multiple objects in an image sequence. The segmentation technique requires only a small number of training samples from a user in order to model the multidimensional probability distribution function (PDF) of the image characteristics at each region. The user-supplied training samples are tracked over time, and the PDF of each region can be updated at every frame. All the unlabeled data points are classified to the most likely region, based on the distance between the PDF of each region, and the multidimensional feature value. A certainty of that classification is computed and stored at each sample for additional post-processing and analysis of the segmentation. The usefulness of such a system has immediate applications in the fields of image processing and computer vision.

Image attributes such as color, texture, motion, and position are used to characterize the different regions, and constitute the basis of our multidimensional feature vector at every sample location in the image sequence . A detailed explanation of a PDF estimation technique known as expectation maximization (EM) explains the basis of how the PDF of each feature is calculated. Improvements to the basic EM algorithm including deterministic annealing are also tested. Varying the feature calculation methods, feature combinations, PDF estimation techniques, and tracking of user-supplied training for several sequences establish the basis of our experiments and results. The experiments presented in this thesis show how the segmentation result is usually robust with respect to the method of calculating the features, and the result is typically better when using all features instead of any one feature exclusively.

Thesis Supervisor: V. Michael Bove, Jr.
Thesis Commitee: W. Eric L. Grimson
Paul Viola

This research has been supported by the Television of Tomorrow and Digital Life Consortia at the Media Laboratory.

In loving memory

of our beloved

Mayer Chalom.

In loving honor

of our beloved

Henriette Chalom.

In loving gratitude

to my beloved

Chalom Family.

Acknowledgments

If we knew what it was we were doing, it would not be called research, would it?

– Albert Einstein.

There have been many times in this research process that I did not know what I was doing nor did I know where my work was taking me. I'd like to thank the many people who have shown faith in me and my work, and who have provided me with keen insight and good direction at times I needed it.

First and foremost, thanks to the MIT Media Laboratory and special thanks to the Television of Tomorrow and Digital Life Consortia for offering me a place to hang out and to do some exciting research on the side. Infinite thanks to my advisor, Mike Bove who has always been there for me with helpful advice.

A profound gratitude to both Paul Viola and Eric Grimson for their wonderful feedback on my research and for being a part of my thesis committee.

A heartfelt thanks to Julie Sung and to John Dakss whose dedicated work of each of them played a direct role in my research; Julie for her assistance in running experiments and gathering data, and John for being the best proofreader/editor I ever had.

Thanks to the many students, faculty, and staff in the Garden, Vismod, and EECS Department. I appreciate all the impromptu conversations with Nuno, Giri, Matt, Shawn, and Bill who have given me general help on both theoretical topics and practical programming savvy. I am forever indebted to Henry and Dan G. who could always fix any computer problem I had (and I've had many!). Stefan thanks for your Isis wizardry. Mike Massey thanks for the Gymnast sequence, fun chats, and disc tossing. Thanks to Roger and Tom S. for being really cool office mates – and to everyone in the garden, just for being.

Thanks to many people in Vismod who have incessantly proven to be a great resource – Roz Picard (who's taught me since she was a grad student), Kris, Martin, Tom, Fang, Lee, Tony, and others for your help on specific and general topics. A special thanks to Sumit, Sunil, and Claudio for their awesome feedback on my presentation.

A graduate education is not just a career choice, it's a lifestyle. I especially want to thank the many friends I've made through the years who will make me miss this kind of lifestyle. Friends from Ashdown, MIT Hillel, Temple Beth Shalom, The Ear, BUDA, and others. And to Sabrina for giving me support and comfort when I needed most.

Last and certainly most, I dedicate this thesis in loving gratitude to my family! Their endless support and faith has been extremely reassuring. Despite the long time and distance between us, they continue to be my primary resource for inspiration. I hope to give back to science and to the world even a tiny fraction of what I've received from them. Henriette, Marc, Esther, Stella, Elie, Yvonne, René, Lili, Mando, David, Leor, Mayer, Henriette, Talia, Spence, Anabelle, Marc Alain, Mayer Max, René Emanuelle, Gabrielle Elie, Mayer Marc, Albert Elie, I love you all! And as my father would say, "*Baruch HASHEM!*"

Contents

List of Figures

List of Tables

Chapter 1

Introduction

People have an innate ability to decompose a natural image scene into objects that constitute the scene. This innate ability is not restricted to a specific type of scene, and it can also be used to recognize objects never seen before. Computers, on the other hand, are still at a primitive level when it comes to understanding a wide range of natural image scenes or image sequences. Most of the progress in automatic image analysis over the years has been either in detecting specific objects/widgets, or in solving a specific task. Both of these analysis require a good prior model or expectation of the input image to be analyzed. The term **image segmentation** refers to decomposing a scene into regions corresponding to objects. Thus in a segmented image, every pixel is labeled as belonging to a region.

One reason why a general image segmentation process can be extremely difficult to realize is because for a given natural scene there can often be two different perceptions on what the "true" decomposition should be. These differences usually depend on the observer and on what the decomposition will be used for. In this thesis we will demonstrate a design that is useful for decomposing image sequences into regions corresponding to real objects. The design process is robust enough to handle almost any natural image sequence, and it is flexible enough to allow for different decomposition, based on a user's definition of what the objects are in the scene.

The goal of our research is to gain a semantic understanding of natural image scenes, and to be able to decompose a general image sequence into regions correspond-

ing to natural objects. In order to meet these goals, we need to answer questions like: What objects are in the scene? Where are these objects in the current frame? And how are the objects moving over time?

Figure 1-1 illustrates frames 1 and 10 from an example image sequence as well as one possible desired segmentation of those frames. The current method of performing this kind of segmentation is quite time consuming and requires detailed manual labor. Our goal is to create a tool that would eliminate most of the manual labor yet also capture the objects sought by the user using minimal user interaction. Figure 1-2 illustrates the location of representative sample points for each region provided by the user. These user-provided points are called **training points** and they are useful in determining statistically how similar the regions are to each other and to each sample point in the image sequence. Based on numerous experiments on several image sequences we have discovered that, although we do not get exactly the same segmentation in our automatic process as the manual segmentation, we do get a surprisingly good segmentation considering we only employ a statistical analysis of simple estimates of image attributes (such as color, motion, texture, and position) at each region of the training data to determine the classification of each pixel in the entire image sequence.

There is both a theoretical and practical motivation for designing and automatic or semi-automatic image segmentation algorithm. Current image compression techniques tile images into arbitrary squares or regions that often do not correspond to real-world objects. The experimental research presented in this thesis will hopefully begin to help determine theoretical bounds on how well we can distinguish objects from each other while preserving their natural structure, based on information contained in the image sequence and based on a minimal amount of user training. The practical side for seeking a solution to our problem is fueled by widespread applications in machine vision and image processing. Applications such as object-oriented video editing, region-based image compression, autonomous vehicle navigation, teleshopping, and more all share a fundamental principle that they must all be able to work on a general class of images, They can not be limited to work on images that

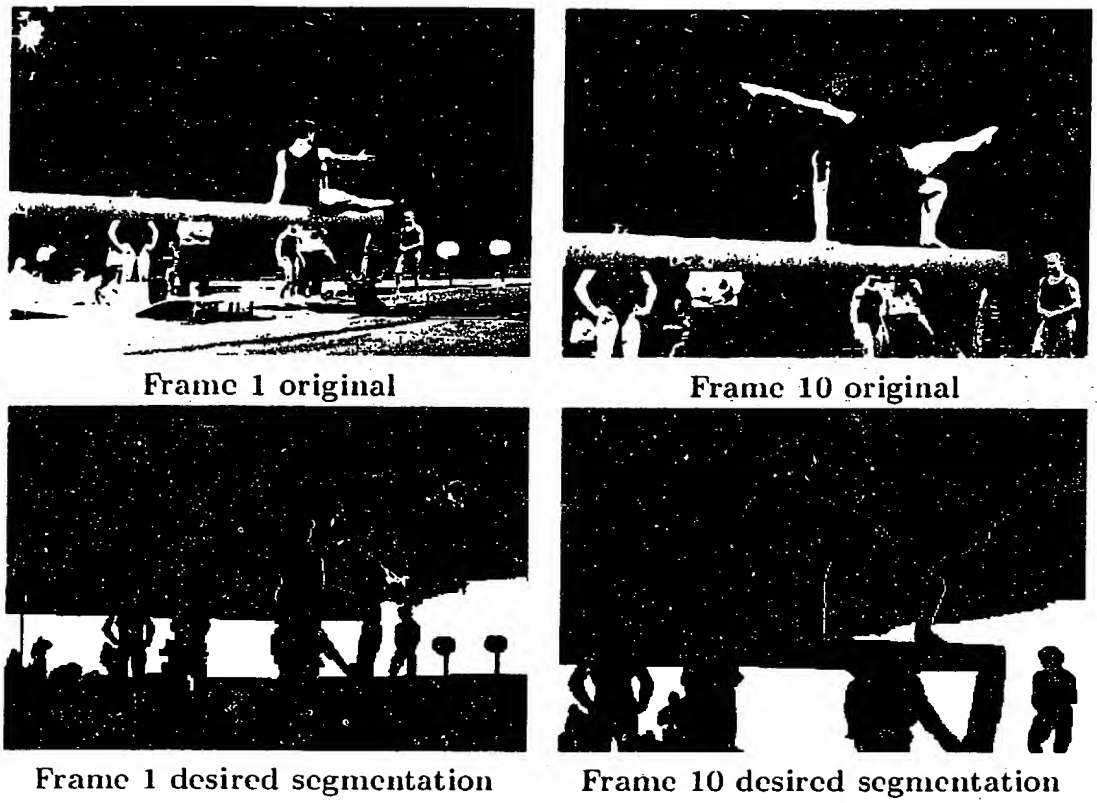


Figure 1-1: Example desired segmentation of GYM sequence.

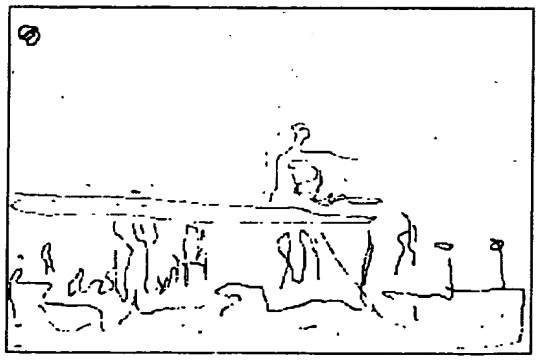


Figure 1-2: Example representative points from each region provided by user.

fit one prior model describing what the expected input image will look like.

There are several novel concepts introduced in our approach to finding an image segmentation solution. First and foremost, regions are user specified, specifically by means of acquiring training data. The benefits of allowing the user to provide training data include: the algorithm is designed to work on a general class of images. We are not limited to detecting widgets, or measuring how similar two faces are, etc. A second advantage is that no prior model is needed for the algorithm to work. The training data in a sense serves as a substitute to the prior model, and therefore no major pre-processing is required. A third advantage is that the segmentation result is adaptive based on the user training. So different users can decide to segment out the same scene in different manners simply by providing the appropriate training data. A fourth advantage of the training data is that it allows for quick minimal user-interaction. So segmentation from the training data is much less time consuming from the user's perspective, compared to the manual segmentation shown in Figure 1-1.

A second novel concept introduced in our image segmentation process is that every sample point in the image sequence is labeled to a region. Other image labeling routines work by tiling an image data base into block-based regions. Selected blocks are labeled by a user as car, grass, sky, etc. And the remaining blocks in the data base are compared to the user-labeled blocks and classified to the most similar type. As a result of employing a block based algorithm, a synthetic block boundary is imposed, and he natural boundary of the object is lost. Finally, a third concept in our segmentation approach is that we can apply the algorithm to image sequences, not just to still images.

Some of our main advantages are also are biggest liabilities. Specifically, since the algorithm is designed to work on a general class of images, there is no prior model, and as a result, the quality of the segmentation depends heavily on the training and how much variance there is between regions. Also, because the algorithm is designed to segment an entire sequence, given only training points in the first frame, there is an incumbent burden on our tracking process. The **tracking** process refers to how the location of the training points is determined from frame to frame.

Other major challenges in our approach include the fact that our goal is not defined quantitatively. There is no good quantitative metric that measures how well structure of an object has been preserved. One reason why this metric is lacking is because the performance evaluation can be subjective. This subjectivity is precisely one of the features our segmentation process exacerbates. Namely, the process allows for different users to segment out the same scene in different manners. While explore the problem predominantly from an experimental approach, there is an ominous realization that the problem is both under constrained and ill posed from a theoretical approach. Despite these challenges there are many contributions that our segmentation experiments offer, even for sake of learning what these shortcomings are to such an open ended approach of the image understanding problem.

1.1 Implementation Overview and Related Papers

I would like to give a very brief overview of the implementation strategy of our segmentation procedure. We are given a video sequence as our input, along with training data points from the first frame in the image sequence. The training data is supplied by the user and corresponds to representative samples from each region. Given only these two input signals, the segmentation implementation can be broken down into four stages.

In the first stage, the input video sequence is used to extract multidimensional characteristics or attributes everywhere in the image sequence. These **image attributes** sometimes called **features** describe the sequence in terms of its motion, color, texture, and position at each sample point in the image sequence. In the second stage, we use motion information to try to estimate the location of the training data at each successive frame. This process is referred to as **tracking**, and is useful because the training data is only supplied in the first frame. Since we are trying to segment multiple frames of an image sequence, the tracking data can be used to determine the location of representative samples from each region at future frames. In the third stage of our segmentation implementation, we use the location of the training

and tracking data, combined with the values of the multidimensional image features to estimate a probability density function (PDF) at each region, for each feature, treating the features as a random variable. Finally, in the fourth stage we use the PDF estimates of all features per each region, to determine which region each sample point in the image space belongs to. Since we have already computed the value of all the features at every pixel in the first stage, we can use this multidimensional feature value to determine the most likely region that each sample point belongs to given the PDF estimation of each region calculated in the third stage. A block diagram of the implementation, as well as more details, is found in Chapter 3.

The research presented in this thesis builds upon many other previous works in the fields of image processing and pattern recognition. Rather than go through a description of other works, I refer you to papers that will hopefully give you some breadth and depth in several categories. References related to segmentation include [49], [82], [68], [1], [4], [5], [27], [23], [52], [57], [40], [85], [86], [2], [3], [51], [21], [47], [22], [96], [11], [61], [19], [39], [28], [29], [98], [15], and [41]. References related to probability estimation and pattern recognition include [87], [60], [92], [93], [37], [54], [75], [32], [58],[33], [59], [43], [83], [100], [95], [31], [46], and [45]. References related to motion estimation include [10], [16], [9], [84], [68], [17], [18], [5], [53], [62], [34], [12], [36], [48], [73], [26], [56], [14], [76], [2], [74], [24], [25], [69], [64], and [63]. References related to texture analysis include [88], [80], [89], [66], [81], [38], [7], [79], and [44]. References related to compression include [53], [50], [73], [30], [97], [26], [72], [77], [6], [71], [74], [21], [61], and [91]. Finally, references related to other image analysis topics include [67], [14], [13], [70], [42], [90], [20], [78], [35], [99], and [63].

1.2 Thesis Overview

Chapter 1 (* you are here *) provides an introduction to the problem, and gives an overview of references to related works. Chapter 2 provides a theoretical basis for understanding probability theory, and probability density function estimation, as well as a brief introduction to pattern recognition and an outline of various classification

methods. Chapter 3 discusses in greater detail each of the stages of the segmentation process introduced in Section 1.1. A theoretical basis and mathematical model is established to support the implementation of the different components of the system. Chapter 4 examines the practical issues of implementing the system. The programs that implement the algorithms are described in detail, and the parameter options of running these programs are discussed. Chapter 5 displays a list of experiments and tests that were done on sequences for which we have manually generated a "ground truth" corresponding to the true or desired segmentation result. Quantitative and qualitative results are shown for these sequences. Also, results were shown on sequences for which the true segmentation is not known. Chapter 6 discusses a few possible applications (such as object editing, interactive video authoring, and compression) for which our images segmentation toolbox could be used. Note that the applications shown are a by product of the global design of the image segmentation process. It was not the main focus of the thesis to solve any one of the applications discussed in Chapter 6, but rather, the main purpose of the research was to experimentally design a general solution to the image segmentation problem. The general segmentation process would require additional details catered to the specific needs in order to implement a fully robust solution to each of those applications.

Chapter 2

Probability Estimation and Classification

An understanding of the key issues in the fields of probability estimation and classification is necessary for one to gain a deeper appreciation for the segmentation process described in Chapters 4 and 5. This chapter will review basic probability theory, as well as elementary tools of probability density estimation, and classification algorithms. These two fields form the foundation of pattern recognition, applicable to many areas of study, including image segmentation discussed in this thesis.

This chapter serves as an overview of important topics pertaining to probability theory, and the reader is encouraged to refer to relevant textbooks for a more in depth review. For a review of basic probability and probability theory, refer to books by Drake, 1988, or Shanmugan and Breipohl, 1988. For a review of classification algorithms and estimation, refer to Therrien, 1989, or Duda and Hart, 1973.

2.1 Basic Probability Theory

The study of probability and probability theory offers useful tools to analyze and understand nondeterministic processes. A **nondeterministic process** (also known as a **random experiment**) is one in which the outcome of a particular measurement or experiment is unknown to an observer. The set of possible outcome values of the

random experiment is called a **sample space**. An **event** refers to a collection of points (or areas) that are a subset of the sample space. A **random variable** (RV) is a variable whose precise value is not known *a priori* to the observer, and is defined by a function which assigns a value of the random variable. Each value of of the RV corresponds to each sample point in the sample space of possible outcomes of performing the random experiment. Each trial of the random experiment is said to generate an **experimental value** of a random variable.

If our experiment is flipping a coin N times ($N > 0$), the sample space of this nondeterministic process does not explicitly have a numeric value. For example, if $N = 1$ there are two possible outcomes; "heads" or "tails", neither of which is a numeric value. We generally define a random variable corresponding to some numeric calculation generated by the sample space. For example we can define h as the total number of heads resulting from the N coin tosses. While there are 2^N possible outcomes of the experiment, there are only $N + 1$ possible values for h. The RV, h, is said to take on experimental values, h_0, where h_0 is in the range $\{0, 1, ..., N\}$.

If the range of experimental values comes from a set of discrete numbers (as in the example of the coin toss experiment), the RV is called a **discrete** random variable. Otherwise, if the range of experimental values comes from a continuous set, the RV is called a **continuous** random variable. A function describing the likelihood of values of a discrete random variable x is called a **probability mass function (PMF)**, and is written as $p_x(x_0)$. In the coin toss example, the RV is discrete, since h lies in the range of integers 0 to N. When $N = 3$, h is defined at $h_0 = \{0, 1, 2, 3\}$, and $p_h(h_0) = \{\frac{1}{8}, \frac{3}{8}, \frac{3}{8}, \frac{1}{8}\}$.

A roll of a six sided die is another example of a random experiment with a discrete random variable. In this case, however, the outcome of the dice roll is a number. Generally, we express the probability that an event x_i can occur from the set of possible outcomes Ω_x, as either $P(x = x_i)$, $P_x(x_i)$, $P_x(x)$, or $P(x)$. In the dice roll experiment we can avoid explicitly defining the event space and implicitly make the connection that the event x_i corresponds to the outcome that x takes on the experimental value $x_0 = i$. The range of x is $\{1, 2, 3, 4, 5, 6\}$, and the PMF, $p_x(x_0) =$

$\{\frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}\}$, in that range.

Before we begin to explore continuous random variables, let us take a look at two basic properties of discrete random variables and probability distributions.

$$0 \leq P_x(x_i) \leq 1 \tag{2.1}$$

$$\sum_{x_0 \epsilon \Omega_X} p_x(x_0) = 1 \tag{2.2}$$

Equation 2.1 states that the likelihood of any event occurring must be bound by 0 and 1. Equation 2.2 states that the sum of the probabilities for all possible experimental values x_0 must be equal to 1. The value of the probability of a discrete value can be obtained either from a physical model or it can be estimated experimentally. In our earlier example of a coin flip, assuming the coin is a fair coin, the physical model tells us that heads is equally likely as tails, and therefore (assuming no other possible outcome) each event outcome has a probability value of 0.5. Alternatively, we can flip the coin N times and define $P_x(x = heads) = \frac{H}{N}$, where H is the total number of heads out of the N tosses. It follows from Equation 2.1 that $P_x(x = tails) = 1 - \frac{H}{N} = \frac{N-H}{N}$.

In the continuous domain, the concept of a likelihood value at a particular experimental value x_0 is not as concrete as in the discrete domain. For example, if we describe the experiment of choosing a real number anywhere in the range between zero and one, there are an infinite number of possible outcomes. The likelihood that the experimental value is exactly equal to x_0 is zero. Therefore, in the continuous domain, an event (or outcome) is said to occur if the experimental value lies in the range $x_a < x < x_b$, for $x_a < x_b$. The likelihood of such an event is defined by

$$P(x_a < X < x_b) = \int_{x=x_a}^{x_b} p_X(x)dx, \tag{2.3}$$

where

$$p_X(x) = \lim_{\delta \to 0} \frac{P(x < X < x + \delta)}{\delta} \tag{2.4}$$

The likelihood of the continuous random variable x is described by its likelihood

function $p_X(x)$. This likelihood function is known as a **probability distribution function** or PDF, and it is the continuous analog to the PMF. The property described in Equation 2.2 extends to the following equation in continuous time:

$$\int_{x=-\infty}^{\infty} p_X(x)dx = 1 \tag{2.5}$$

And the property described in Equation 2.1 is modified as

$$0 \leq p_X(x_0) < \infty \tag{2.6}$$

The remaining discussion in this thesis will deal almost exclusively with real-number continuous random variables. Section 2.2 discusses in detail how to estimate $P(x)$ given observation data or experimental outcomes of the random variable x. Section 2.3 discusses how to classify the observation data into clusters or regions, where all (or most) of the data from each region correlates well to one probability distribution function estimate.

A **joint probability** density, or multidimensional probability density function, is a function of two or more variables. We write $p_{A,B}(a,b)$ to denote that the probability is a function of two random variables, A, and B. Either of the unconditional one dimensional PDF, $p_A(a)$ or $p_B(b)$ can be calculated from the joint probability function by integrating over the other variable. Thus

$$p_A(a) = \int_{b=-\infty}^{\infty} p_{A,B}(a,b)db \tag{2.7}$$

$$p_B(b) = \int_{a=-\infty}^{\infty} p_{A,B}(a,b)da \tag{2.8}$$

The **conditional probability** of a two random variable system, is a one dimensional function of one variable, given the known value of the other random variable. We write $p_{A|B}(a|b)$ to refer to the probability of random variable a given that the value of B is known. Mathematically, a conditional probability is related to the joint

probability as follows:

$$p_{A|B}(a|b) = \frac{p_{A,B}(a,b)}{p_B(b)} \tag{2.9}$$

$$p_{B|A}(b|a) = \frac{p_{A,B}(a,b)}{p_A(a)} \tag{2.10}$$

Combining the two pairs of equations above, we get the following relations:

$$p_{A|B}(a|b) = \frac{p_{A,B}(a,b)}{\int_{a=-\infty}^{\infty} p_{A,B}(a,b)da} \tag{2.11}$$

$$= \frac{p_{B|A}(b|a)p_A(a)}{p_B(b)} \tag{2.12}$$

$$= \frac{p_{B|A}(b|a)p_A(a)}{\int_{a=-\infty}^{\infty} p_{B|A}(b|a)p_A(a)da} \tag{2.13}$$

Equations 2.12 and 2.13 are known as Bayes' Rule, in two slightly different forms.

2.2 Probability Function Estimation

This section explores how one would begin to estimate the underlying probability density function (PDF) of a random variable. **Statistics** deals with methods for making decisions based on measurements or observations collected from the results of experiments. The coin toss example showed us that we can have an *a priori* model and a PMF estimate describing the behavior of the random variable, namely that heads and tails are equally likely for a fair coin. Alternatively, if we don't know whether the coin is fair (or if we simply don't have an underlying model of the random variable, due to a more complex scenario), we can examine empirical or statistical evidence from numerous trials of the experiment measuring the experimental value of the RV at each trial to help decide between different models, and/or to determine the best PDF estimate given a specific model.

There are two classes of PDF estimation techniques: **parametric** and **nonparametric**. For parametric estimation of $p_X(x)$, we assume we have a model for the

shape of the PDF, and we use the experimental outcome to determine the best parameters to fit that assumed shape. For example, the shape of the density could be uniform, Gaussian, or multimodal Gaussian. The observation data samples are used to find the "optimal" values of the parameters that quantitatively describe the model. A uniform distribution has a constant height and is characterized simply by its range. The height of the distribution is solved for by imposing the constraint in Equation 2.5. A Gaussian distribution is fully described by its mean and variance. Finally, a multimodal Gaussian distribution is by the weight, mean, and variance of each mode, as well as the number of modes. Equations 2.14 and eqn:multimodal-Gaussian describe the PDF of a Gaussian and multimodal Gaussian distribution. In general, for nonparametric estimation of $p_X(x)$, we assume we do not know the underlying shape of the PDF, thus we let the "data speak for itself." Specifically, the relative frequency of the raw values of the data is used to calculate a PDF estimate.

2.2.1 Nonparametric Estimators

Nonparametric estimators are used when the underlying shape of the probability density function is not known. In the context of **supervised segmentation** problems, (discussed in further detail in Section 2.3) nonparametric estimators typically fall into two categories: In the first category the PDF estimation of each region is calculated explicitly based on the model and the values of the **training samples** from each region. Section 3.4.2 discusses in greater detail how this estimation is done. In the second category of nonparametric estimators, a segmentation boundary is applied directly by classifying each sample of the **observation data** set based upon its "proximity" to each region of the training data. An example method from this second category is known as **K Nearest Neighbor** and is described in [95].

2.2.2 Parametric Estimators

A parametric estimator is used if the shape, though not the parameters, of the underlying distribution is known or assumed. For example, if the PDF shape is assumed

(or known) to be uniform, then there are only two parameters to estimate, namely the minimum and maximum value the observation data can take on. If the distribution shape is assumed to be unimodal Gaussian (or normal), then only the mean and variance needs to be estimated. If the underlying distribution is assumed to be a sum of Gaussians, (also known as a **mixture density of Gaussians** or a **multimodal Gaussian distribution**) then there is a vector of parameters that must be estimated. Namely, the number of modes (M) the mixture density is comprised of, as well as the mean, variance and weight (m_i, σ_i, w_i) of each mode ($1 \leq i \leq M$) in the mixture density. It turns out that for many of the image attributes calculated in Chapter 3, a mixture density model is an appropriate estimate of the underlying distribution. The next section describes in detail one specific algorithm to estimate the parameters of a mixture density, for a given value of M, corresponding to the number of modes in the model.

Expectation Maximization for a Mixture Density

This section will examine a method known as the **Expectation Maximization** or EM algorithm, which is used to estimate the parameters of a PDF. In particular, we will assume the shape of the PDF is a mixture of Gaussians, and that we have N data samples chosen arbitrarily from the distribution. Section 2.2.3 discusses in detail a numerical example programmed in Matlab, that simulates the basic EM algorithm as well as some additional practical modifications.

The expectation maximization or EM algorithm is an iterative solution which can be used to find the "optimal" parameters of a mixture density, for a specific value of M, the number of modes in the mixture density. The basic outline for the algorithm is as follows.

1. Begin with an estimate of the parameters (e.g. the mean, variance, and weight of each mode) of the mixture density.

2. For each data sample point, calculate the *Conditional class probability* or the likelihood that a sample belongs to mode i, (for $1 \leq i \leq M$), given the data, and

the parameters of the mixture density. This step is known as the **expectation** or E-step, since we are calculating the expectation that each sample belongs to each mode.

3. Given the conditional class probability, we can update our estimate of the mean, variance and weight of each mode of the mixture density. This step is known as the **maximization** or M-step, since we are calculating the maximum likelihood values of the parameter vector.

4. Repeat steps 2 and 3 until there is little or no change in the mixture density estimate.

Since the EM algorithm gives us the optimal PDF for a particular value of M, we generally repeat the EM algorithm with a different value of M at each iteration. Thus, by varying M from 1 through 8, for example, we have 8 different "optimal" PDF estimates. Each one is optimal for an M mode mixture density PDF model. A second step described later is used to decide which of the models is most likely.

Let us first look closer at the EM algorithm to estimate a one dimensional PDF $P(x)$ for a fixed (assumed known) number of modes M. We define the observation data $\mathbf{x}$ as a vector of N one dimensional sample points. If we define θ as our parameter vector containing the mean, weight, and variance of each Gaussian component in the mixture density, then we need to solve for $3M$ parameters, assuming the data is one dimensional. We solve for the θ parameter vector by introducing an intermediate variable i that represents which mode an observation data sample x belongs to.

Recall that a one dimensional Gaussian or normal PDF has the form

$$P(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{-(x-\bar{x})^2}{2\sigma^2}\right) \tag{2.14}$$

where $\bar{x}$ and σ^2 represent the mean and variance. As a shorthand notation sometimes we simply write $P(x) = N(x; \bar{x}, \sigma_x^2)$, and the PDF of a mixture of Gaussians has the

form

$$P(x) = \sum_{i=1}^{M} \frac{w_i}{\sqrt{2\pi}\sigma_i} \exp\left(\frac{-(x-m_i)^2}{2\sigma_i^2}\right) \tag{2.15}$$

where w_i, m_i, σ_i^2 represent the weight, mean, and variance of the i^{th} component.

Starting with an initial estimate of θ, we wish to compute the likelihood that a random variable x would be drawn from a Gaussian PDF corresponding to the i^{th} mode of the mixture density. Using Bayes' Rule (see Equation 2.12), this can be expressed as

$$P(i|x;\theta) = \frac{P(i,x;\theta)}{P(x;\theta)} \tag{2.16}$$

Sometimes $P(i|x;\theta)$ is written simply as $P(i|x)$ and the θ parameter is inferred. Other times it can also be written as $P(x\epsilon\omega_i|x)$, meaning the likelihood that a sample x is an element of the i^{th} mode or class (ω_i) given the observation data (and the parameter estimate).

For a given value of θ, the numerator of Equation 2.16 reverts to

$$P(i,x) = P(x|i) \cdot P(i). \tag{2.17}$$

$P(i)$ is simply w_i (one of the parameter values contained in θ) and represents the weighting coefficient of the i^{th} mode. $P(x|i)$ is simply the value of a Gaussian PDF with mean m_i and variance σ_i^2 (also parameter values contained in θ) evaluated at each data sample x.

The denominator of Equation 2.16 is simply a multimodal Gaussian PDF, and conceptually it can be thought of as a normalizing function. $P(i|x)$ measures the likelihood a particular observation sample would be drawn from a Gaussian PDF with parameters m_i and σ_i^2, for each $1 \leq i \leq M$. The numerator ensures that $\sum_{i=1}^{M} P(i|x) = 1$. We can also describe the denominator of Equation 2.16 by the follow-

ing relationship:

$$P(x;\theta) = \sum_{j=1}^{M} P(x; m_j, \sigma_j^2) \cdot P(j) \tag{2.18}$$

which is exactly the PDF for a mixture of Gaussians described in Equation 2.15.

In the first half of each iteration of the EM algorithm we solve Equations 2.15 through 2.16 to get a better estimate of our PDF based on the current estimate of our parameter vector θ This is known as the E-step, or the expectation each sample belongs to each mode. In the second half of each iteration we use the current estimate of the PDF to get a better estimate of our parameters which will in turn be used at the next iteration. This is known as the M-step, or finding the maximum likelihood parameters of each mode. To solve for the M-step given N observations of x and assuming the number of modes M is known, then for $1 \leq i \leq M$

$$w_i = \frac{1}{N} \sum_{k=1}^{N} P(i|x)|_{x=x_k} \tag{2.19}$$

$$m_i = \frac{\sum_{k=1}^{N} P(i|x)|_{x=x_k} \cdot x_k}{\sum_{k=1}^{N} P(i|x)|_{x=x_k}} \tag{2.20}$$

$$\sigma_i^2 = \frac{\sum_{k=1}^{N} P(i|x)|_{x=x_k} \cdot x_k^2}{\sum_{k=1}^{N} P(i|x)|_{x=x_k}} - m_i^2 \tag{2.21}$$

The new parameter values calculated from Equations 2.19 through 2.21 are used in the next iteration to determine a better estimate of $P(i|x)$. Eventually when there is little or no difference between the parameters from one iteration to the next, the algorithm comes to an end and the best estimate of the underlying multimodal PDF is given by Equation 2.15 using the appropriate parameters calculated from the EM algorithm.

Entropy and Kullback Liebler Distance

A proof that the EM algorithm converges is found in [43, 92] Rather than go into the details of the proof, I want to show how the entropy calculation (useful in the proof) measures the "quality" of the PDF estimation. In a physical system, **entropy** refers to the amount of disorder in a system. In a probabilistic system, the entropy is defined as

$$H_p = -\int_{-\infty}^{\infty} p(x)\log(p(x)) \tag{2.22}$$

and qualitatively it measures the state of disorder or uncertainty in the system. Often times the entropy of a system is said to be inversely proportional to the information in the system. Thus by minimizing the entropy we gain more certainty or more information.

Given N observations on a random variable x, suppose we wish to implement the EM algorithm on our observation data to estimate the true underlying PDF that could have generated these sample points. At each iteration of the EM algorithm, the effective **Kullback Liebler distance** or **entropy distance** between the PDF estimate (based on the observation) and the true (unknown) underlying PDF is implicitly minimized with respect to the parameter vector θ. Let us define $H_{\hat{p}}$ as the effective distance that is minimized in each EM step, where

$$H_{\hat{p}} = -\frac{1}{N}\sum_{k=1}^{N}\log(\hat{p}(x))|_{x=x_k}. \tag{2.23}$$

$H_{\hat{p}}$ is not quite the entropy of $\hat{p}(x)$ (where entropy is defined by Equation 2.22), but rather it is effectively a cross entropy, or entropy distance between $p(x)$ and $\hat{p}(x)$. (Note that in Equations 2.23 through 2.28 $p(x)$ refers to the true underlying PDF of the RV x and $\hat{p}(x)$ refers to the PDF estimate.) We can derive Equation 2.23 beginning with the Kullback Liebler (KL) distance or entropy distance between $p(x)$ and $\hat{p}(x)$.

$$D(p(x)||\hat{p}(x)) = \int_{-\infty}^{\infty}\log\left(\frac{p(x)}{\hat{p}(x)}\right)p(x)dx \tag{2.24}$$

$$= \int_{-\infty}^{\infty} p(x)\log(p(x)) - \int_{-\infty}^{\infty} p(x)\log(\hat{p}(x)) \quad (2.25)$$

$$= -H_p - \int_{-\infty}^{\infty} p(x)\log(\hat{p}(x)) \quad (2.26)$$

$$= -H_p - E[\log(\hat{p}(x))] \quad (2.27)$$

$$\approx -H_p - \frac{1}{N}\sum_{k=1}^{N} \log(\hat{p}(x))|_{x=x_k} \quad (2.28)$$

The approximation listed in Equation 2.28 comes from

$$E[f(x)] = \int_{-\infty}^{\infty} p(x)f(x)dx \quad (2.29)$$

$$= \lim_{N\to\infty} \frac{1}{N}\sum_{k=1}^{N} f(x)|_{x=x_k} \quad (2.30)$$

$E[f(x)]$ corresponds to the mean or average value of $f(x)$, where x is a random variable, $f(x)$ is any well behaved function of x, and x_k is the k^{th} (of N observations) of the RV. If N is a finite number of observations then we don't know the true mean of $f(x)$, so the best estimate for the mean of $f(x)$ is

$$\hat{E}[f(x)] = \frac{1}{N}\sum_{k=1}^{N} f(x)|_{x=x_k} \quad (2.31)$$

As a quick example suppose $f(x) = x$, then $\hat{E}[f(x)] = \hat{E}[x]$, which corresponds to both the best estimate of the mean of $f(x)$ as well as the best estimate of the mean of x. And the best estimate of the mean is simply,

$$\hat{E}[x] = \frac{1}{N}\sum_{k=1}^{N} x_k \quad (2.32)$$

This is sometimes known as the **statistical mean** of x, namely the sum of the observations divided by the number of observations. Often times the statistical mean is not exactly equal to the true mean, since the true underlying PDF of x is unknown. However, the statistical mean is the "best" estimate of the true mean given the N finite observation samples.

In Equation 2.28, H_p corresponds to the entropy of $p(x)$ (as defined by Equa-

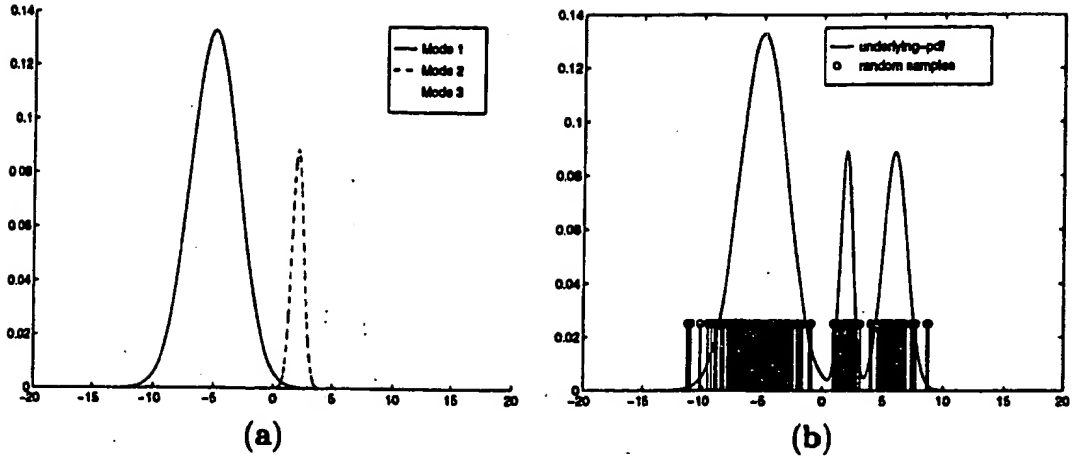


Figure 2-1: (a) Three slightly overlapping Gaussians, with means $\{-5, 2, 6\}$, standard deviations $\{2, 0.5, 1\}$, and weights $\{\frac{2}{3}, \frac{1}{9}, \frac{2}{9}\}$. (b) The mixture of Gaussian PDF is shown, as well as 300 observation points selected at random from the above PDF.

tion 2.22) and it is a constant, so minimizing the Kullback Liebler distance in Equation 2.24 reverts to minimizing the second term in Equation 2.28. And this second term in Equation 2.28 that we minimize is identical to Equation 2.23.

2.2.3 Numeric Implementation of the EM Algorithm

This section will show the results of the basic EM algorithm implementation described in Section 2.2.2, and we will also examine two modifications that should lead to improved estimates. All three methods will use the same set of random data. Figure 2-1a shows three normal distributions, with means $\{-5, 2, 6\}$, standard deviations $\{2, 0.5, 1\}$, and a scaling of $\{\frac{2}{3}, \frac{1}{9}, \frac{2}{9}\}$. Figure 2-1b shows the sum of these three Gaussians, (*i.e.* the true underlying PDF) as well as 300 sample observations points chosen randomly from the prescribed underlying mixture density. For demonstration purposes, each random sample is depicted by a stem, with a constant height chosen arbitrarily. The task is to recover the true underlying PDF shape, or the parameters that describe the true PDF, given only the data sample values at the N observation points. The solution has two parts: in the first part, we run the EM algorithm with $M = 1, 2, \ldots, 8$ modes, thereby ending up with 8 possible PDF estimates. In the second part of the solution, we use a discriminating calculation to find which of the

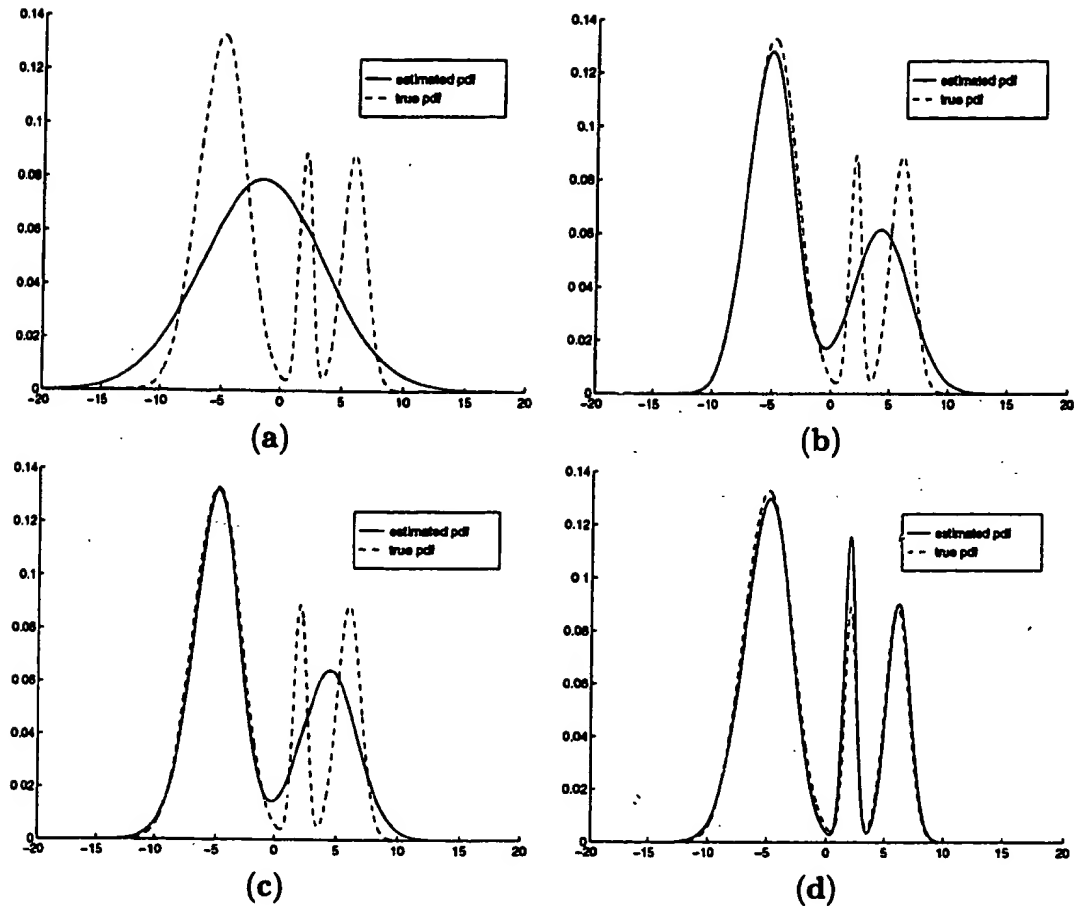


Figure 2-2: (a)-(d) Multimodal PDF estimation using basic EM algorithm; number of modes, M, varies from 1 through 4.

8 possible estimates is the "best" estimate.

Basic EM Algorithm

Using Matlab, the basic EM algorithm described in Section 2.2.2 was implemented. The resulting PDF estimate for $M = 1, 2, \ldots, 8$ modes is shown in Figure 2-2a-h. Table 2.1 shows the parameter estimates using the basic EM algorithm for $M = 1, 2, \ldots, 8$. Two adjustments to the basic EM algorithm are described later in the section. The resulting estimate based on those implementations are shown in Figures 2-3 and 2-4, and the corresponding parameter estimates are shown in Tables 2.2 and 2.3,

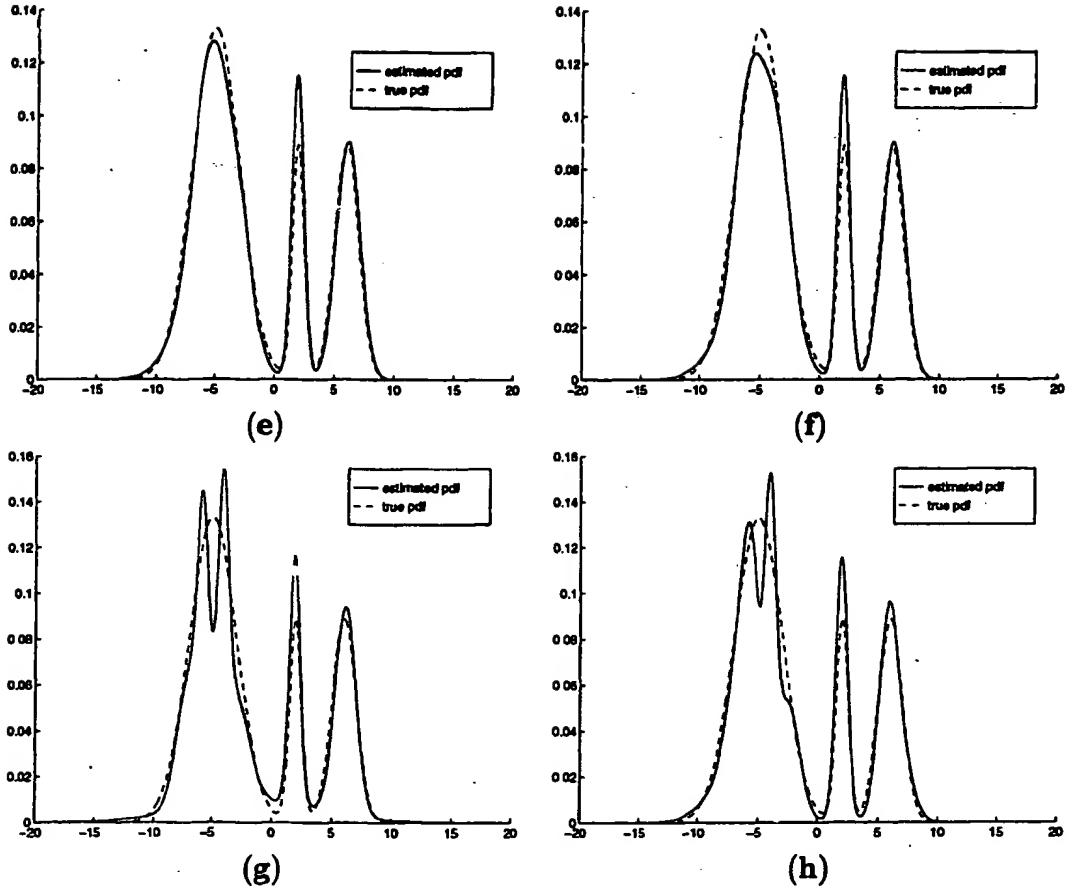


Figure 2-2: (e)-(h) Multimodal PDF estimation using basic EM algorithm; number of modes, M, varies from 5 through 8.

M	Mean	Std. Dev.	Weight
1	-1.57	5.04	1.00
2	-5.16 4.22	1.92 2.49	0.62 0.38
3	-5.27 -4.87 4.42	2.32 1.60 2.29	0.31 0.32 0.37
4	-5.60 -4.53 1.94 6.11	2.11 1.69 0.49 0.99	0.32 0.32 0.14 0.22
5	-5.97 -5.72 -3.52 1.94 6.11	2.19 1.22 1.37 0.49 0.99	0.21 0.21 0.21 0.14 0.22
6	-6.25 -6.17 -3.94 -3.92 1.94 6.11	2.25 1.06 1.47 1.47 0.49 0.99	0.16 0.16 0.16 0.16 0.14 0.22
7	-7.06 -5.80 -4.15 -3.11 -1.68 1.93 6.10	0.99 0.51 0.48 1.17 5.51 0.46 0.86	0.14 0.14 0.14 0.13 0.13 0.13 0.20
8	-6.68 -6.53 -5.60 -4.03 -2.52 1.94 5.78 6.44	2.15 0.98 0.68 0.45 1.01 0.49 0.73 1.10	0.13 0.13 0.13 0.13 0.12 0.14 0.11 0.11

Table 2.1: Estimated parameters of multimodal PDF estimation using the basic EM algorithm; number of modes, M varies from 1 through 8.

Initializing EM Parameters

Examining the PDF estimation example of Figure 2-2, we notice that the estimate when $M = 3$ modes does not produce a good fit to the true underlying PDF. It is not until $M = 4$ modes that the estimated PDF shape begins to take on the shape of the true PDF. In fact, if we used 200 instead of 300 observation points (though not shown here), it would require $M = 5$ modes, to get the correct underlying PDF shape. One reason for this sub-optimal estimation is because the EM algorithm does not perform a global search over the parameter space, but instead uses a gradient descent method given the initial guess of the parameter vector. This creates a possibility for the resulting PDF estimate to land at a local minimum rather than a global minimum.

Therefore, the first adjustment that is made to the basic EM algorithm is refining the initial guess of the parameter vector θ. In the basic EM algorithm implementation (result shown in Figure 2-2), the means in the initial iteration are equally spaced between $m_1 - \sigma_1$ to $m_1 + \sigma_1$, for $M > 1$, where m_1 and σ_1 are the mean and standard deviation of the PDF estimate for $M = 1$ mode. In our example, $m_1 = -1.57$ and $\sigma_1 = 5.04$, so when $M = 2$ the initial estimate of the mean is $\{(-1.57 - 5.04), (-1.57 + 5.04)\}$. When $M = 5$ for example, the initial guess of the

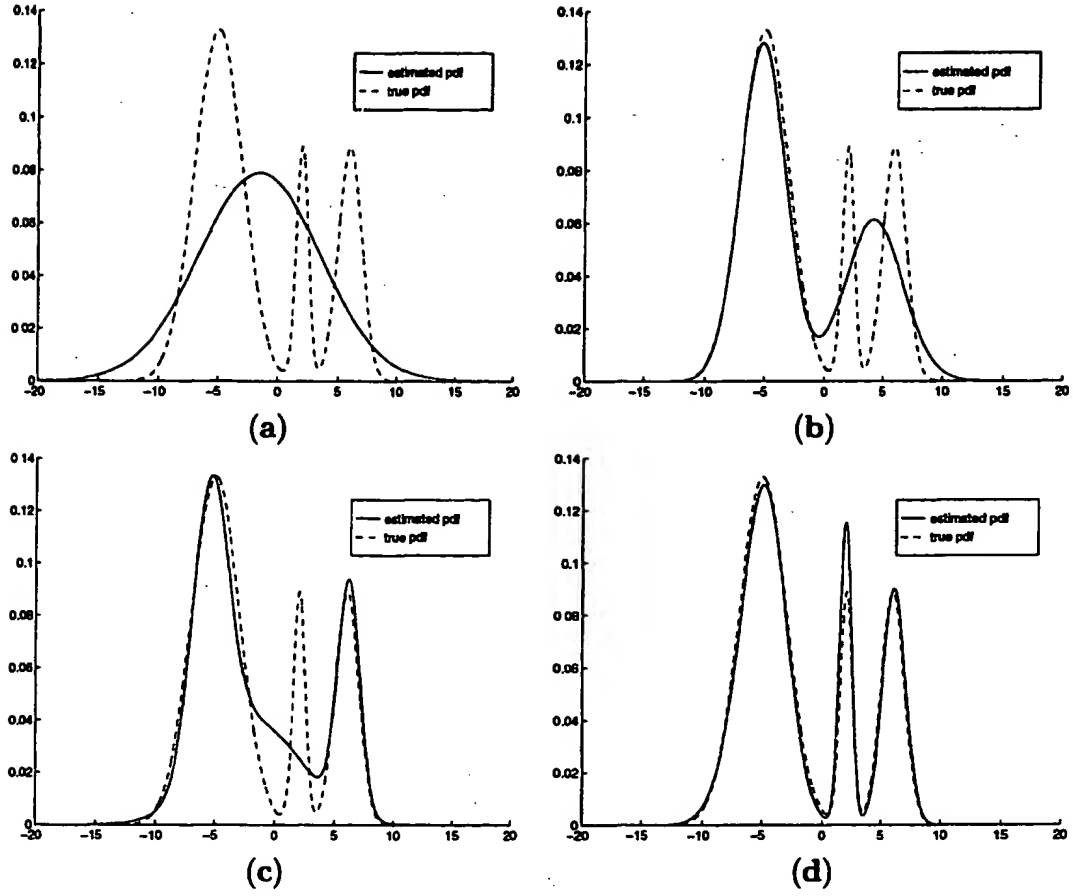


Figure 2-3: (a)-(d) Multimodal PDF estimation using EM algorithm, and parameter initialization modification; varying number of modes between 1 through 4.

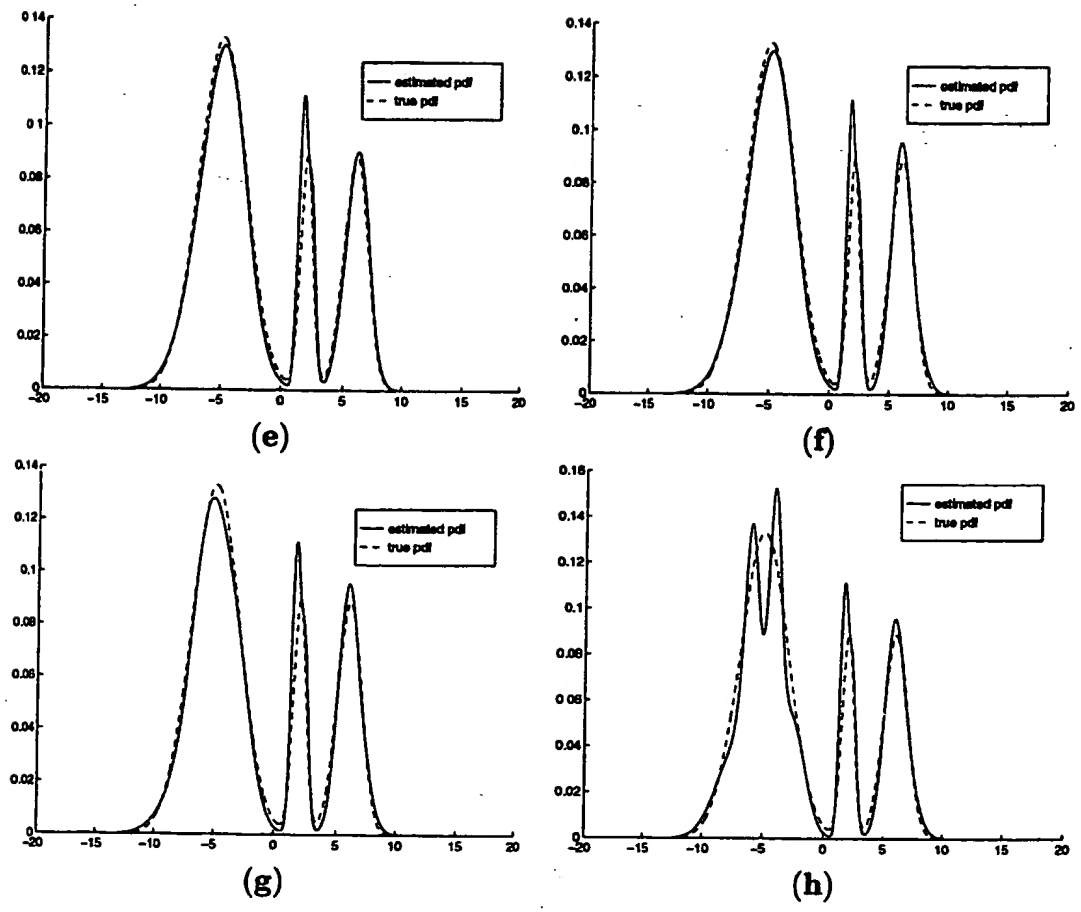


Figure 2-3: (e)-(h) Multimodal PDF estimation using EM algorithm and parameter initialization modification; varying number of modes between 5 through 8.

M	Mean	Std. Dev.	Weight
1	-1.57	5.04	1.00
2	-5.16 4.22	1.92 2.49	0.62 0.38
3	-5.40 -1.69 6.14	1.50 3.96 0.94	0.40 0.39 0.21
4	-5.59 -4.54 1.94 6.11	2.11 1.69 0.49 0.99	0.32 0.32 0.14 0.22
5	-5.61 -4.52 1.58 2.30 6.11	2.10 1.69 0.30 0.38 0.98	0.32 0.32 0.07 0.07 0.22
6	-5.62 -4.52 1.58 2.30 5.73 6.50	2.10 1.69 0.30 0.38 0.73 1.05	0.32 0.32 0.07 0.07 0.11 0.11
7	-6.00 -5.71 -3.50 1.58 2.30 5.73 6.50	2.17 1.22 1.36 0.30 0.38 0.73 1.06	0.21 0.21 0.21 0.07 0.07 0.11 0.11
8	-7.08 -5.96 -4.15 -2.74 1.58 2.30 5.73 6.50	1.64 0.65 0.51 1.13 0.30 0.38 0.73 1.06	0.17 0.17 0.15 0.15 0.07 0.07 0.11 0.11

Table 2.2: Estimated parameters of multimodal PDF estimation using the EM algorithm with parameter initialization modification; number of modes, M varies from 1 through 8.

mean is $\{(-1.57-5.04),(-1.57-\frac{5.04}{2}),(-1.57),(-1.57+\frac{5.04}{2}),(-1.57-5.04)\}$. The initial guess of the weight of each mode in the basic algorithm is $1/M$, and the initial guess of the standard deviation of each mode is σ_1.

In Figure 2-3, we use the initialization scheme of the basic algorithm only for $M = 2$ modes. However for $M = 3$ modes we perform the basic EM algorithm search twice, each time with a different set of initialization parameters. In the first search, our estimate for the mean is $\{(m_1^{(2)}-\sigma_1^{(2)}),(m_1^{(2)}+\sigma_1^{(2)}),(m_2^{(2)})\}$; the estimate of the weight is $\{(w_1^{(2)}/2),(w_1^{(2)}/2),(w_2^{(2)})\}$ and the estimate of the standard deviation is $\{(\sigma_1^{(2)}),(\sigma_1^{(2)}),(\sigma_2^{(2)})\}$. In the second search (for $M = 3$ modes), we split the second mean and weight parameters. Namely, our estimate for the mean is $\{(m_1^{(2)}),(m_2^{(2)}-\sigma_2^{(2)}),(m_2^{(2)}+\sigma_2^{(2)})\}$; the estimate of the weight is $\{(w_1^{(2)}),(w_2^{(2)}/2),(w_2^{(2)}/2)\}$ and the estimate of the standard deviation is $\{(\sigma_1^{(2)}),(\sigma_2^{(2)}),(\sigma_2^{(2)})\}$. The notation $m_k^{(j)}$ represents the mean of k^{th} mode ($k \leq j$) resulting from the PDF estimate when $M = j$ modes.

More generally, for each $M > 1$ modes we run the EM algorithm $M-1$ times, each time starting with a different initial estimate of the parameters. Specifically, we use the resulting PDF estimate from the $M-1$ mode model as our initial guess

for the M mode model, except we split the $m_j^{(M-1)}$ mean into $m_j^{(M-1)} - \sigma_j^{(M-1)}$, and $m_j^{(M-1)} + \sigma_j^{(M-1)}$, for $j = 1, \ldots, M-1$. The corresponding initial estimate of the weight and standard deviation is $w_j^{(M-1)}/2$, and $w_j^{(M-1)}/2$, and $\sigma_j^{(M-1)}$, and $\sigma_j^{(M-1)}$. Thus, the basic EM algorithm is run $M-1$ times, and we need to compare the performance of the $M-1$ resulting PDF estimates. This comparison is done by observing which of the estimates best fit the N observation samples, which is the same as minimizing the Kullback Liebler distance in Equation 2.23.

For example, if $M = 3$ modes, we would run the EM algorithm 2 times. Using the values from Table 2.2, the first time the initial parameter estimate of the mean would be $\{(-5.16-1.92), (-5.16-1.92), (4.22)\}$; on the weight, $\{0.31, 0.31, 0.38\}$ and on the standard deviation, $\{1.92, 1.92, 2.49\}$. The second time, the initial parameter estimate of the mean would be $\{(-5.16), (4.22-2.49), (4.22+2.49)\}$; on the weight, $\{0.62, 0.19, 0.19\}$ and on the standard deviation, $\{1.92, 2.49, 2.49\}$.

Deterministic Annealing

Even with the improvement of using a more robust search of initial parameters, we notice from Figure 2-3 that the PDF estimate resulting from $M = 3$ modes is still not accurate enough. Therefore a modification known as **deterministic annealing**, (abbreviated as DA or EMDA) introduces a hidden temperature variable which is used to avoid local minima of the Kullback Liebler distance defined in Equation 2.23.

The term "annealing" comes from statistical physics, where materials, such as glass or metal, are first heated to very high temperatures and then slowly cooled. as the temperature is gradually lowered, the material is maintained at thermal equilibrium. In our EMDA implementation we modify Equation 2.16 to include a temperature parameter, T, as follows:

$$P(i|x) = \frac{w_i \cdot (N(x; m_i, \sigma_i^2))^T}{\sum_{j=1}^{M} w_j \cdot (N(x; m_j, \sigma_j^2))^T}. \tag{2.33}$$

T is an implicit temperature parameter, and $N(x; m_i, \sigma_i^2)$ represents a Gaussian with mean m_i and variance σ_i^2. In the basic EM algorithm, as well as the robust search of

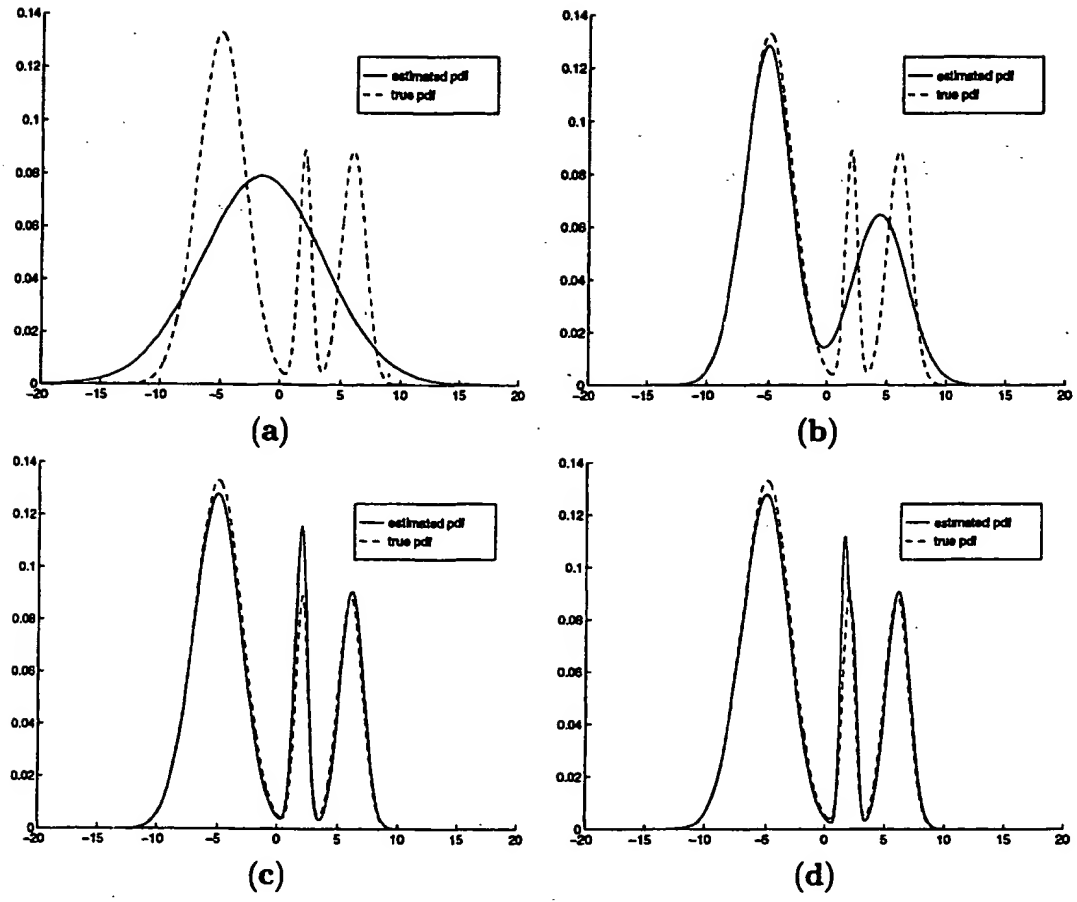


Figure 2-4: (a)-(d) Multimodal PDF estimation using EM algorithm, with parameter initialization modification, and deterministic annealing; varying number of modes between 1 through 4.

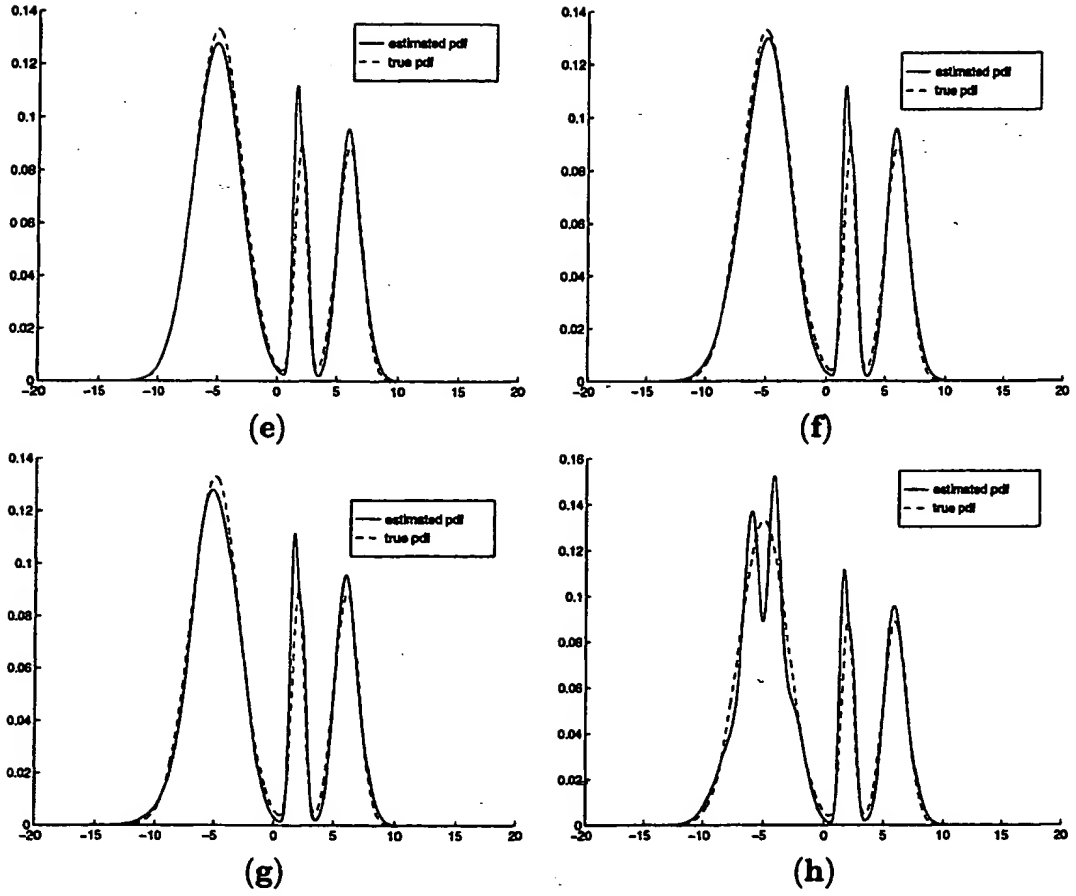


Figure 2-4: (e)-(h) Multimodal PDF estimation using EM algorithm, with parameter initialization modification, and deterministic annealing; varying number of modes between 5 through 8.

initialization parameters modification, we implicitly set $T = 1$, and so the equation is exactly that of Equation 2.16. However, with the deterministic annealing approach, we begin with a high temperature estimate and find the optimal EM parameters for that temperature. As we gradually decrease the temperature level, our initial estimate of the PDF parameter θ is taken from the result of the EM algorithm at the previous temperature level; we also add a small random offset to the means m_i in order to keep the algorithm from remaining "trapped" at the same parameters at each temperature level.

An alternate way of thinking about the deterministic annealing process is by minimizing the **free energy** equation (F in Equation 2.34) with respect to the conditional class likelihood variable in our expectation step of EM . Recall that the E-step is where we calculate the conditional class likelihood for each sample and determine how likely a sample belongs to each mode of the multimodal PDF.

$$F = E - TH \tag{2.34}$$

where E corresponds to the energy component, H corresponds to the entropy of the conditional class probability, and T is the temperature parameter, that starts large and is gradually decreased. Minimizing the free energy is the same as jointly maximizing the entropy, H, and minimizing the energy, E.

First let us define the energy component characterized by E as:

$$E = \sum_{k=1}^{N} \sum_{i=1}^{M} g_i(x_k) \frac{(x_k - m_i)^2}{\sigma_i^2} \tag{2.35}$$

where $g_i(x_k)$ is the conditional class likelihood (similar to $P(i|x)|$ defined in Equation 2.16) that the k^{th} data point belongs to the i^{th} mode or class. M is the number of modes of our multimodal Gaussian, N is the number of data points, x_k is the k^{th} data point, m_i is the mean of the i^{th} mode, and σ_i is the standard deviation of the i^{th} mode. In statistical physics, the σ_i is often set to a constant among all M modes, and the term is often dropped. (As a side note, if the $g_i(x_k)$ coefficients are restricted

to be either 0 or 1, then we are in effect solving the K-means algorithm described in Section 2.3.1 and in [95].) Minimizing the energy will find only a local

Next let us define the entropy component characterized by H as:

$$H = -\sum_{k=1}^{N}\sum_{i=1}^{M} g_i(x_k)\ln(g_i(x_k)) \tag{2.36}$$

H measures the entropy of the conditional class probability function g, it does not measure the entropy of our multimodal PDF estimation. By starting with a high T parameter and gradually decreasing it we are not restricted to a local minimum of the energy component described in Equation 2.35. More detailed explanation of solving EM and deterministic annealing via a free energy equation can be found in [58], [59], and [100]. Note that in our implementation, we don't assume equal variances in all modes, and instead we approach EMDA as described in eqn:da-cond-class-prob. If we assume a constant variance of $\sigma_k = 1$ for all modes, both methods should produce the same EM solution provided they follow the same temperature schedule.

The result of applying the deterministic annealing modification to the basic EM algorithm is shown in Figure 2-4, and the corresponding parameter values are shown in Table 2.3. With this modification, we now have a very good estimate for $M = 3$ modes.

Cross Validation

We have shown how deterministic annealing combined with the EM algorithm can result in a good PDF estimate. However, we have run the estimation procedure for 1 through 8 modes, and we now need to decide between the different models. Table 2.4 shows the entropy of the best PDF estimate for all 8 models from each of the 3 methods discussed. Information theory tells us that as we increase M (the number of modes in our model), the entropy of our best PDF estimate decreases. So if we have have N data observations, the PDF estimate which minimizes the Kullback Liebler distance between the data and the PDF estimate is one where we have $M = N$ modes, with the mean at each mode exactly at the data value, the weight equal to

M	Mean	Std. Dev.	Weight
1	-1.57	5.04	1.00
2	-5.11 4.38	1.95 2.30	0.63 0.37
3	-5.07 1.94 6.11	1.98 0.50 0.98	0.63 0.14 0.22
4	-5.07 1.59 2.30 6.11	1.98 0.30 0.38 0.98	0.63 0.07 0.07 0.22
5	-5.07 1.58 2.30 5.73 6.50	1.98 0.30 0.38 0.73 1.05	0.63 0.07 0.07 0.11 0.11
6	-5.62 -4.51 1.58 2.30 5.73 6.50	2.09 1.69 0.30 0.38 0.73 1.06	0.32 0.32 0.07 0.07 0.11 0.11
7	-6.00 -5.71 -3.50 1.58 2.30 5.73 6.50	2.17 1.22 1.36 0.30 0.38 0.73 1.06	0.21 0.21 0.21 0.07 0.07 0.11 0.11
8	-7.08 -5.96 -4.15 -2.74 1.58 2.30 5.73 6.50	1.64 0.65 0.51 1.13 0.30 0.38 0.73 1.06	0.17 0.17 0.15 0.15 0.07 0.07 0.11 0.11

Table 2.3: Estimated parameters of multimodal PDF estimation using the EM algorithm with parameter initialization modification, and deterministic annealing; number of modes, M varies from 1 through 8.

Modes	Method I	Method2	Method 3
1	3.0371	3.0371	3.0371
2	2.7913	2.7913	2.7895
3	2.7872	2.7644	2.6473
4	2.6453	2.6453	2.6439
5	2.6433	2.6420	2.6413
6	2.6428	2.6395	2.6395
7	2.6478	2.6375	2.6375
8	2.6277	2.6286	2.6286

Table 2.4: Performance comparison of entropy.

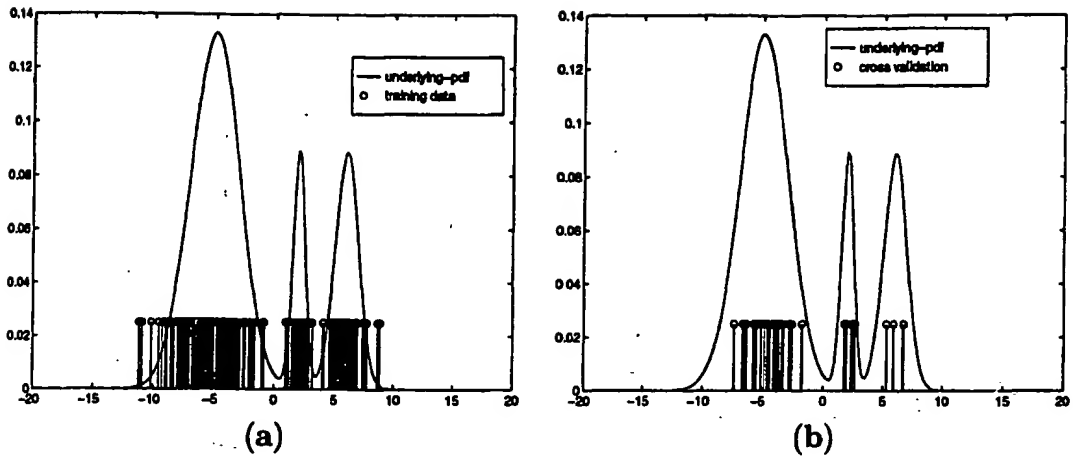


Figure 2-5: (a) 85% of total observation data shown in Figure 2-1 used for training data to calculate PDF parameters. (b) 15% of observation points used as test data for cross validation entropy measurement to find best model between $M = 1$ through 8 modes.

M	Mean	Std. Dev.	Weight
1	-1.43	5.21	1.00
2	-5.24 4.51	2.02 2.30	0.61 0.39
3	-5.20 1.90 6.11	2.05 0.52 1.00	0.62 0.14 0.25
4	-5.20 1.52 2.27 6.11	2.05 0.28 0.42 1.00	0.62 0.07 0.07 0.25
5	-5.20 1.52 2.27 5.69 6.54	2.05 0.28 0.42 0.72 1.06	0.62 0.07 0.07 0.12 0.12
6	-5.72 -4.68 1.52 2.27 5.69 6.53	2.15 1.81 0.28 0.42 0.72 1.06	0.31 0.31 0.07 0.07 0.12 0.12
7	-6.19 -5.90 -3.51 1.52 2.27 5.69 6.53	2.20 1.20 1.43 0.28 0.42 0.72 1.06	0.20 0.21 0.21 0.07 0.07 0.12 0.12
8	-7.39 -5.97 -4.18 -2.68 1.52 2.27 5.69 6.53	1.52 0.56 0.48 1.15 0.28 0.42 0.72 1.06	0.17 0.17 0.14 0.14 0.07 0.07 0.12 0.12

Table 2.5: Estimated parameters of multimodal PDF estimation based only on training data ($0.85N$ sample points) using the basic EM algorithm and deterministic annealing; number of modes, M, varies from 1 through 8.

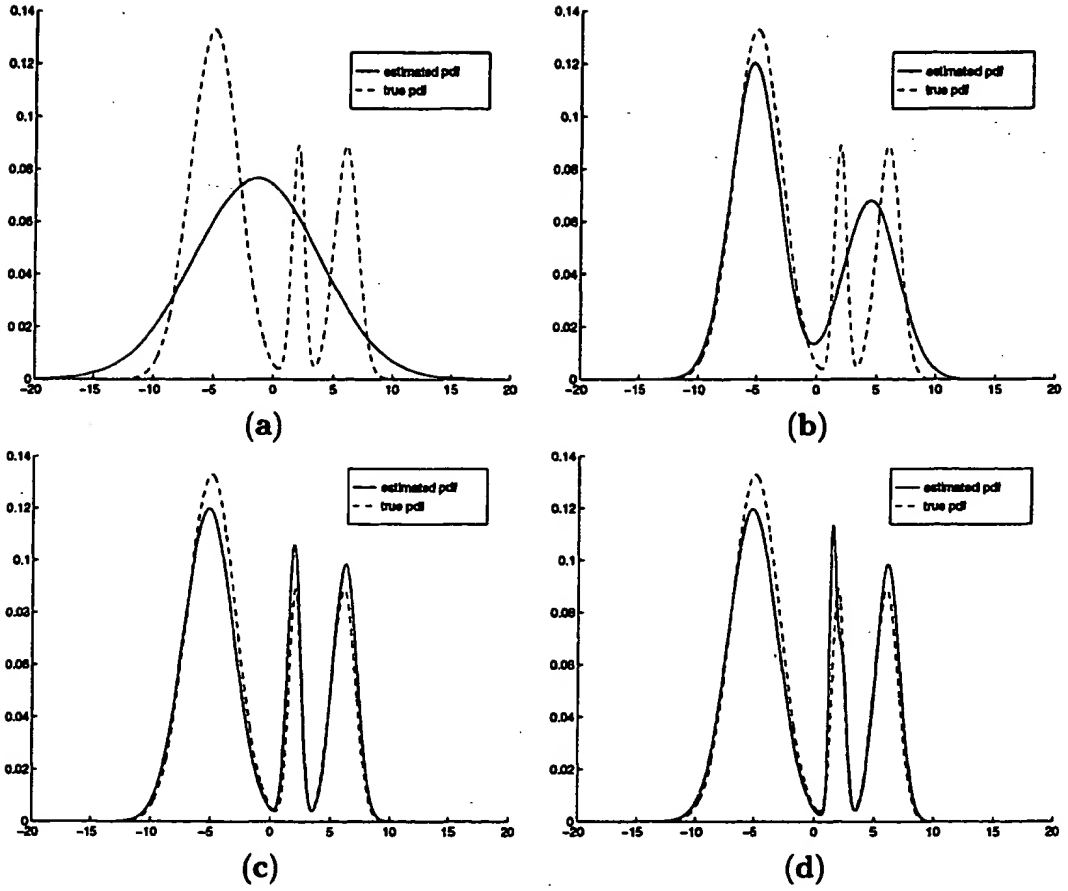


Figure 2-6: (a)-(d) Multimodal PDF estimation based on training data ($0.85N$ sample points) using the basic EM algorithm and deterministic annealing; number of modes, M, varies from 1 through 4.

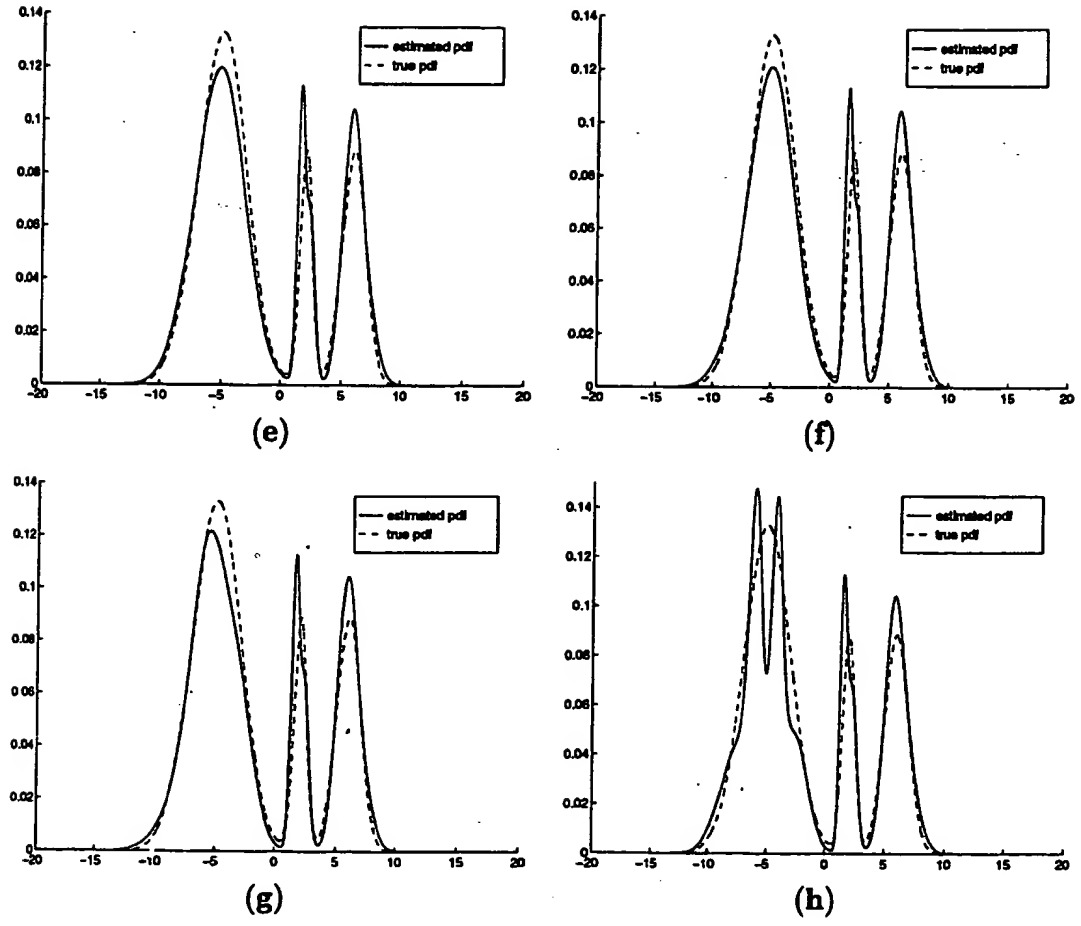


Figure 2-6: (e)-(h) Multimodal PDF estimation based on training data ($0.85N$ sample points) using the basic EM algorithm and deterministic annealing; number of modes, M, varies from 5 through 8.

Modes	Training Data	Cross Validation
1	3.0704	2.8577
2	2.8215	2.6293
3	2.6852	2.4608
4	2.6796	2.4759
5	2.6765	2.4777
6	2.6757	2.4699
7	2.6732	2.4724
8	2.6623	2.4873

Table 2.6: Kullback Liebler distance or entropy measurement of training data points, as well as the cross validation entropy measurement made with the remaining test data points for $M = 1$ through 8 modes. Minimum cross validation entropy occurs at $M = 3$ modes. $N = 300$ total observation samples; $0.15N = 45$ test data points; $0.85N = 255$ training data points.

$\frac{1}{N}$, and the standard deviation of 0. However, that extreme over fit would certainly not be a good estimate for the true underlying PDF from which those N data points were arbitrarily chosen. Therefore choosing the minimum KL distance is not the best criterion to choose between the 8 models, as it will lead to an estimate that over fits the observation data.

One possible method for avoiding over fitting the data, is examining the change in KL distance as the number of modes increases and choosing the estimate that coincides with the greatest change in KL distance. However, when this method was tested, it was found to work best only when the true underlying PDF had $M = 2$ modes. An alternate solution is known as **cross validation**. Cross validation is a method where a fraction of the original observation data samples is kept aside to be used as **test data**, and the remaining data points are used as **training data**. The fraction, F, is typically chosen to be between 10% and 20%. The best PDF estimate for $M = 1$ through 8 modes is calculated from the remaining training data points. The KL distance between the test data points and the estimated PDF for each $M = 1$ through 8 is calculated, and the model with the minimum distance is chosen as the best PDF estimate of the entire observation data.

Figure 2-5(a) shows the training data which came from randomly selecting 85% of

the total data from Figure 2-1(**b**). Figure 2-5(**b**) shows the test data points, namely the remaining 15% of the total data points that are not used to estimate the PDF parameters, but rather are used in the cross validation testing process. Figure 2-6 shows the resulting PDF estimates for all 8 multimodal models, using only the training data. Note how similar the results are to Figure 2-4 which used all the training data to obtain the PDF parameter estimates. Table 2.5 shows the values of the parameters for each of the models. Finally, Table 2.6 shows the entropy of the training data for each model, *i.e.* the Kullback Liebler distance between the training data and the PDF estimate. It also shows the cross validation entropy, or the KL distance between the test data and the PDF estimate. Notice that the distance is a minimum at $M = 3$, which is the number of modes in the true underlying PDF.

More Examples

Here are four more examples of estimating the PDF of several different true underlying multimodal Gaussians. In all four examples, we use the EM algorithm with deterministic annealing to estimate the best fit to models with $M = 1$ through 8 modes, and we use the cross validation techniques to select between the 8 models. In each example, $N = 300$ total observation points, and we use $0.85N = 255$ of those points for training, and the remaining $0.15N = 45$ of the data points for cross validation. Figures 2-7(**a**),(**d**),(**g**),(**j**) show the true underlying PDF vs. the 255 training data samples for each of the four examples. Figures 2-7(**b**),(**e**),(**h**),(**k**) show the true underlying PDF vs. the 45 training data samples. And Figures 2-7(**c**),(**f**),(**i**),(**l**) show the true underlying vs. the best PDF estimate, where best implies the minimum cross validation entropy.

2.3 Classification Algorithms

In Chapter 3 we will discuss how the PDF estimation techniques described in the previous section are a fundamental step in our image segmentation process. In an **image segmentation** process, we begin with the assumption that each pixel in the

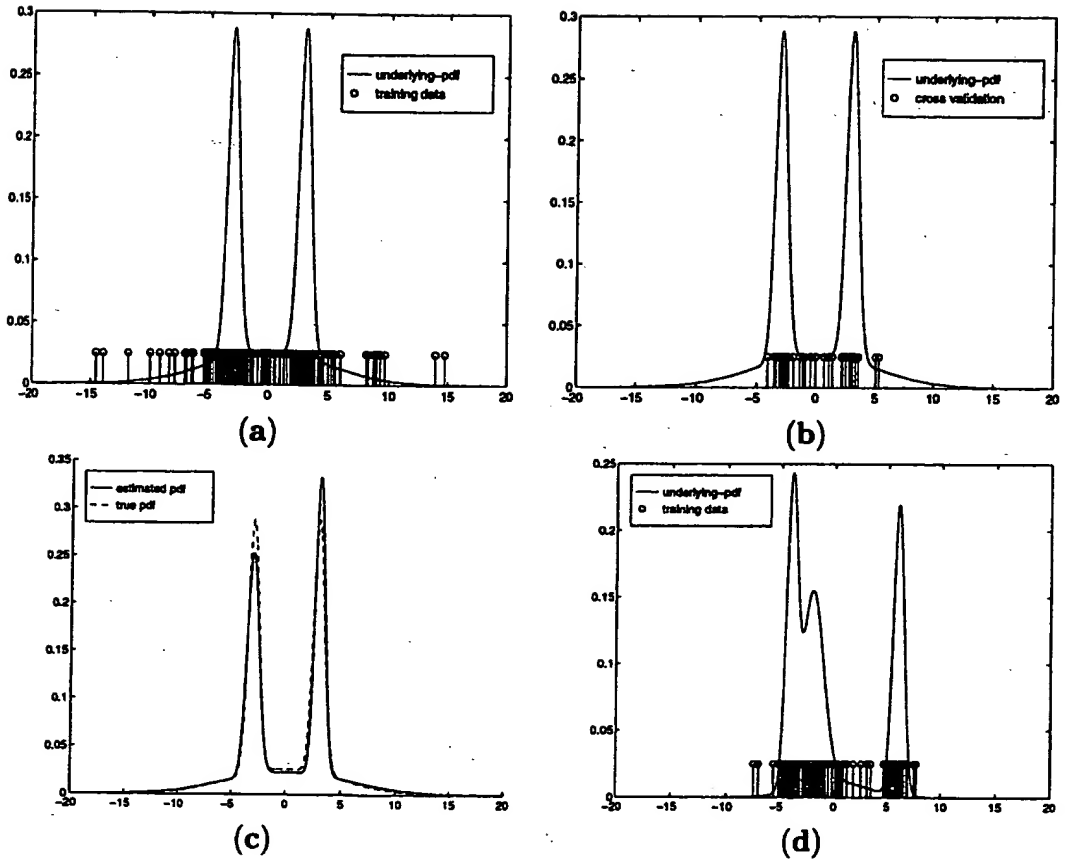


Figure 2-7: (a)-(d) More PDF estimation examples using EM, deterministic annealing, and cross validation.

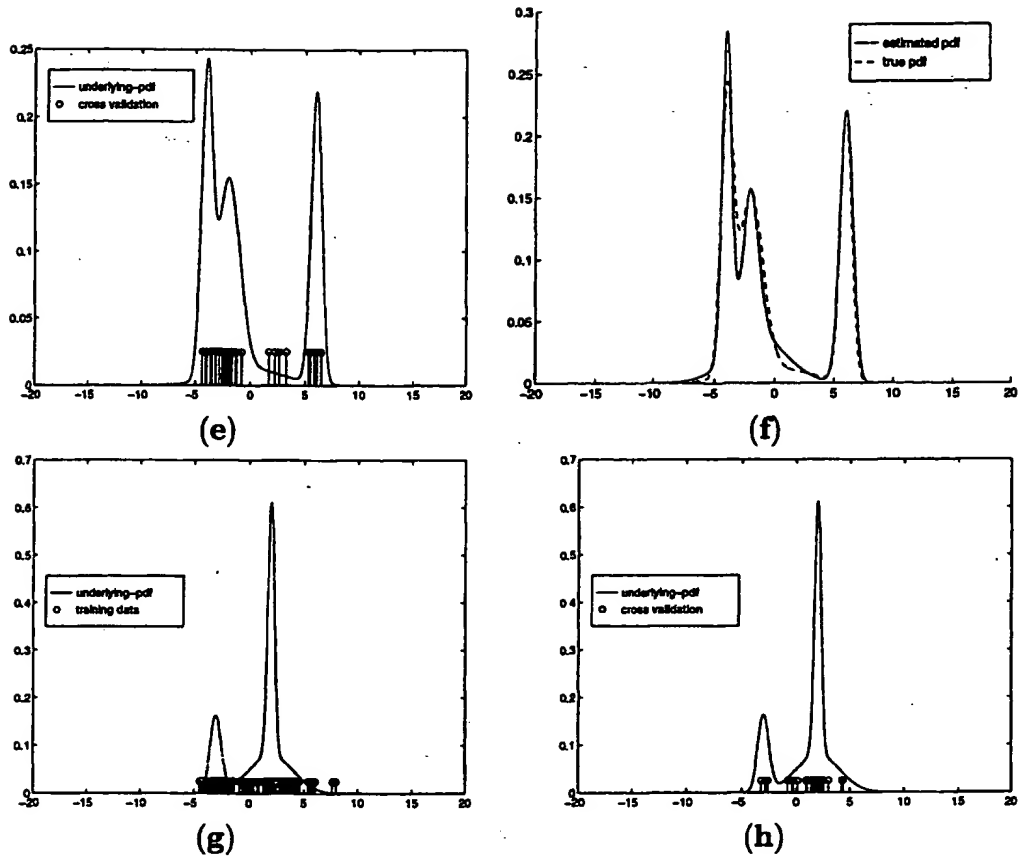


Figure 2-7: (e)-(h) More PDF estimation examples using EM, deterministic annealing, and cross validation.

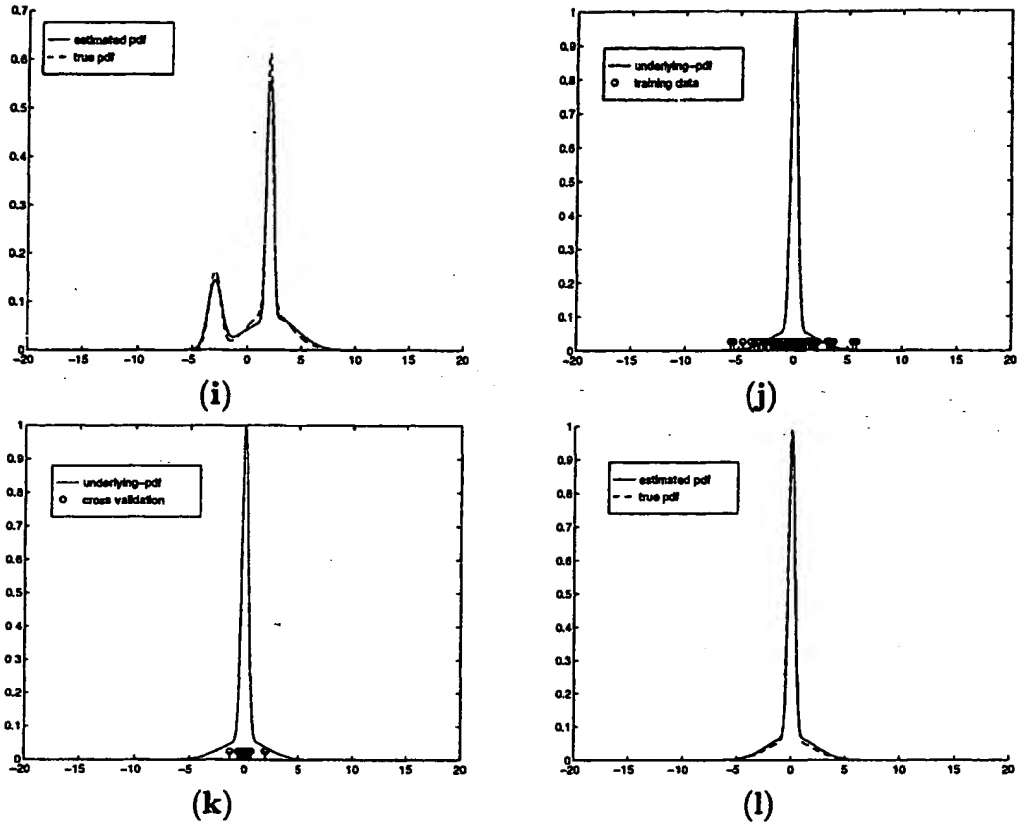


Figure 2-7: (i)-(l)More PDF estimation examples using EM, deterministic annealing, and cross validation.

Fig ref	M	Mean	Std. Dev.	Weight
(a-c) True	3	-3.00, 0.00, 3.00	0.50, 5.00, 0.50	0.33, 0.33, 0.33
(a-c) Est.	3	-3.03, 0.21, 3.06	0.55, 5.68, 0.44	0.32, 0.33, 0.35
(d-f) True	4	-4.00, -2.00, 0.00, 6.00	0.50, 1.00, 3.00, 0.50	0.27, 0.36, 0.09, 0.27
(d-f) Est.	4	-4.00, -2.01, -1.40, 6.04	0.40, 0.69, 2.38, 0.55	0.26, 0.21, 0.23, 0.30
(g-i) True	3	-3.00, 2.00, 2.00	0.50, 2.00, 0.30	0.20, 0.40, 0.40
(g-i) Est.	5	-3.00, 0.71, 1.82, 2.22, 3.35	0.58, 2.12, 0.22, 0.22, 1.78	0.20 0.20 0.20 0.20, 0.20
(j-l) True	2	0.00, 0.00	2.00, 0.30	0.30, 0.70
(j-l) Est.	2	0.06, 0.02	2.07, 0.28	0.38, 0.62

Table 2.7: True and estimated parameters of each of the four examples of Figure 2-7

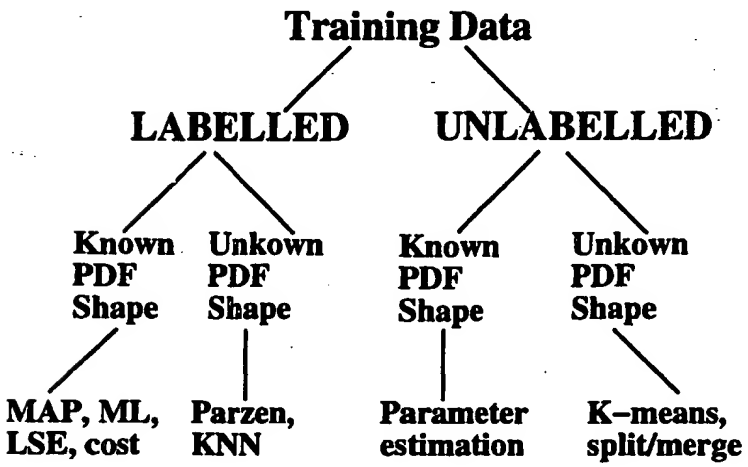


Figure 2-8: This chart shows several examples of both supervised and unsupervised segmentation techniques.

image sequence belongs to a large class or region or group of pixels, usually corresponding to a object, and the goal is to segment or identify these regions, by labeling each pixel. Before describing in detail the classification algorithm used specifically for image segmentation, let us take a look at some general properties of classification or pattern recognition problems.

Figure 2-8 shows a chart naming different parameters in a classification system. Any problem in which we have a large number of data points that need to be labeled or categorized or grouped or classified is called a *classification* problem. Typically the points that make up a group have an aspect of similarity, and the problem is

usually in recognizing or detecting a pattern. Therefore, this class of problems is often referred to as **pattern recognition**.

In designing a classification algorithm, for a given problem, the different parameters (or branches) in the chart are tested. Therefore, we refer to the top node of the chart as the **training data**. Once the classification procedure is designed, the training data is substituted with actual data. If the classification of some of the data is labeled, the classification is known as a **supervised** learning algorithm. Otherwise, if none of the data is pre-classified, the classification algorithm is an **unsupervised** one. Sections 2.3.1 and 2.3.2 will discuss in a bit more detail some general strategies for designing an unsupervised and supervised segmentation algorithms.

2.3.1 Unsupervised Classification

We will talk about unsupervised algorithms first, since in the remainder of the thesis we will be discussing a supervised algorithm for our image segmentation toolbox. In an unsupervised algorithm, recall that none of the data is labeled, all the data points must be categorized into one of R regions.

First, assume that the shape of the PDF for each region of the observation data is known. If the number of regions, R, is known, then the only variables that need to be solved for are the parameters that describe the PDF. For example, if we have data for four regions, and we know that the observed data at each region can be described by a Gaussian PDF, then we only need to solve for the mean and variance describing the PDF of each region. In such a case, an iterative process very similar to the EM algorithm (described in Section 2.2.2) could be used. The only difference is that the EM algorithm described was to solve for one multimodal PDF with four modes, and in our unsupervised segmentation example, we are looking for four Gaussian PDF's, each being unimodal. The problem becomes more involved if R is not known, and/or if the PDF shape is more complex than a unimodal Gaussian.

Now assume that the shape of the PDF for each region is not known. If the number of regions, R, is known, then an algorithm such as K-means (similar to vector quantization), can be used. In this method, an initial estimate of the mean and vari-

ance for each class is chosen. Each data point is classified to the "nearest" region, based on some distance metric, usually Mahalanobis. A new estimate for the mean and variance of each region is calculated based on the current classification. The algorithm iterates at each step refining the classification based on the new mean and variance, followed by updating the mean and variance based on the modified classification. The algorithm ends when there is no change from iteration to iteration. Minor variations on initialization conditions include a randomization of the classification to get an initial mean and variance estimate. Also there may be some minor differences in ending the iteration process.

If the number of regions is not known, (and the shape of the PDF is not known), the K-means algorithm is modified to allow for split and merge conditions. Basically, at each iteration a split condition is tested based on heuristics, such as if the number of samples in a region is too high and/or the variance in that region is too high. A merge condition tests whether the mean between two regions is too low, and/or if the number of samples or the variance is too low. **ISODATA** [95] is an example of such a split and merge algorithm. Problems with these split and merge algorithms are that they are not guaranteed to converge, the final segmentation is extremely sensitive to the initialization parameters, and the number of regions varies greatly from frame to frame when applied to image segmentation. We have previously tested an unsupervised image segmentation algorithm, and the results are discussed in [26].

2.3.2 Supervised Classification

The left side of Figure 2-8 shows the branches associated with labeled or supervised classification algorithms. In each region, there exists a subset of test or labeled training data from the entire data set of which we know the labeling for that test data. Typically, the subset of labeled data can be anywhere from $1 - 10\%$ of the entire data, and the task is to label the remaining unlabeled data as best as possible. One main advantage of using a supervised classification method, is that we implicitly know the number of regions or classes in the data. Another major advantage, is that the labeled data in each region can serve as a model to either explicitly or implicitly

estimate the PDF of each region, and thereby classify the remaining unlabeled data.

Unknown PDF Shape

In the case when the PDF shape for the underlying data is not known, we use methods such as: histograms, Parzen estimators, and K-nearest neighbor (KNN) algorithms. In a histogram method, the PDF of each region is estimated numerically, based on the values of the observed training data, in each region. Histogram methods can often be sensitive to the number of bins, thereby causing inconsistent PDF estimates as the number of bins varies. Often, they are not very robust when the sample size is low, since the raw data may contain "holes" in the observation.

Parzen estimators are a smoothed version of the histogram. Basically, a kernel with a specific shape (uniform, triangular, Gaussian) and width is applied to the data, as an attempt to intelligently fill the "holes" in the raw data. The main difficulty with Parzen estimators is in deciding the width of the kernel. Often a regressive technique is used to calculate the kernel width (as a function of the training data) to "optimally" smooth the data, or minimize its entropy. In theory, a Parzen estimate with a Gaussian kernel should produce the same PDF as the EM algorithm applied to the test data, assuming a multimodal Gaussian PDF model.

A third supervised classification algorithm that is used when the shape of the PDF is not known, is called K-Nearest Neighbor or KNN. The KNN algorithm differs from the histogram and Parzen estimator in that no PDF estimation is required in the KNN method. At each unlabeled data point, we look at the data points within some neighborhood of our feature space, and assign the current point to the same region that the majority of the neighbors are assigned to. Two variations of KNN are the **pooled** method and the **grouped** method. In the pooled version of KNN, the neighborhood size around the unlabeled sample is fixed, and the point as classified to the region corresponding with the largest number of samples within that fixed neighborhood size. In the grouped version of KNN, we begin with a small neighborhood size, and slowly increment the neighborhood size until any one of the classes contains a fixed, predetermined number of samples within the neighborhood.

The unlabeled point is classified to that (first) region encountered.

Here is a quick example of the KNN algorithm: Suppose the observation data points are heights of (adult) people, and the classification consists of two classes, W = Women, and M = Men. Suppose the training data values, in inches, are, $D = \{62, 63, 64, 65, 67, 68, 69, 70, 71, 72, 73, 74\}$. And the corresponding classification for the training data is, $\Omega = \{W, W, W, W, M, W, M, W, M, M, M, M\}$. (Note: These numbers are completely fabricated, to make a point only. They are not indicative of a true distribution.) Now suppose one of our unlabeled data points is $d = 69$. Using the KNN algorithm pooled method, with a neighborhood size of 1 inch, the result would be to classify the point as class W, since 2 labeled points belong to class W, and 1 labeled point belongs to class M within a 1 inch neighborhood. If we increase the neighborhood size to 4 inches, the result would be to classify the point to class M. If we use the KNN algorithm grouped method, and our predetermined number of "votes" for a region is 4, the algorithm would reach a conclusion of classifying the sample to class M since after a neighborhood size of 3 inches, 4 of the neighbors belong to class M. On the other hand, a neighborhood size of 5 inches would be required to encounter 4 neighbors in the W class.

Known PDF Shape

Next we will discuss classification in the case where we know or assume the PDF to be parametric, that is, we assume (or know that) the shape of the PDF can be characterized by a set of parameters. There are two components to solving a parametric supervised classification. The first component is solving for the parameters of the PDF of each region from the training data. The second component is deciding which region each of unlabeled data points belongs to, given the PDF of each region.

This second component is often referred to as **Hypothesis Testing**, since we test to see which class each sample observation point (or vector, in the case of multidimensional observation data) belongs. We define $\bar{y}$ as an observation measurement vector, and we say that $\bar{y}$ belongs to class ω_i. If we maximize $P[\omega_i|\bar{y}]$, then we label the observation point as belonging to class ω_i. From Bayes' rule in Equation 2.16,

maximizing $P[\omega_i|\bar{y}]$ with respect to i, is the same as maximizing $\frac{p_{\bar{y}|\omega_i}(\bar{y}|\omega_i)\cdot P[\omega_i]}{p_{\bar{y}}(\bar{y})}$. Since the denominator is not a function of i, and is always greater than zero, we only need to maximize the numerator above. If we assume we know the PDF at each region, which is denoted as $p_{\bar{y}|\omega_i}(\bar{y}|\omega_i)$, and is the same as the first term in the numerator above. The term $P[\omega_i]$ refers to the *a priori* probability that a sample belongs to class i. Often the prior class distribution is unknown, so these values are all set to be equal (namely to $1/R$, where R is the number of regions).

One can also apply a cost function in such a way that would place a higher penalty on one type of error over another type. For example, if there are two regions, one can measure the total error as

$$E = P[\omega_1|\omega_2] + P[\omega_2|\omega_1] \tag{2.37}$$

where $P[\omega_1|\omega_2]$ means the probability a point is labeled as belonging to ω_1 when it really belongs to ω_2. By applying a penalty or cost function to each error term in Equation 2.37, we are left with minimizing a risk function. The details of which can be found in chapter 3 of [95]. In our segmentation application, we do not apply a non-uniform cost function, so the risk function does not come into play. Section 3.5 reviews this hypothesis testing calculation for our image segmentation application with a little more detail.

The hypothesis testing component works, provided we know the PDF of each region, and optionally, the *a priori* likelihood and cost function associated with the various types of errors. Let us now discuss two methods of estimating the parameters of the PDF.

Let us define $p_{x;\theta}(x;\theta)$ to describe the distribution of a random variable, x, given the parameter θ. If we have n observations of this random variable, x_1,x_2,...,x_n, then we say that $p_{x_1,x_2,\ldots,x_n;\theta}(x_1, x_2, \ldots, x_n; \theta)$ is the joint PDF of x_1-x_n, assuming the observations are independent, identically distributed (i.i.d.) measurements. We

define a **likelihood function** $L(\theta)$ as:

$$L(\theta) = p_{x_1,x_2,\ldots,x_n;\theta}(x_1, x_2, \ldots, x_n; \theta) = \prod_{i=1}^{n} p_{x_i;\theta}(x_i; \theta) \tag{2.38}$$

The value $\hat{\theta}$ that maximizes $L(\theta)$ is called a **maximum likelihood** (ML) estimator. Because many PDF have exponential form, and because the ln function is a monotonic function, then we can find $\hat{\theta}$ that maximizes $\ln(L(\theta))$, and often it is easier to do so.

A second method for parameter estimation is known as **maximum** *a posteriori* or MAP. Here we try to find θ that maximizes

$$p_{\theta|x_1,x_2,\ldots,x_n}(\theta|x_1, x_2, \ldots, x_n) = \frac{p_{x_1,x_2,\ldots,x_n|\theta}(x_1, x_2, \ldots, x_n|\theta) \cdot p_\theta(\theta)}{p_{x_1,x_2,\ldots,x_n}(x_1, x_2, \ldots, x_n)} \tag{2.39}$$

Maximizing the numerator of Equation 2.39 with respect to θ is equivalent to maximizing Equation 2.39, since the denominator is not a function of θ. Again, as with the ML estimation, we assume that the observations are i.i.d. Thus, maximizing Equation 2.39 with respect to θ is the same as maximizing $M(\theta)$

$$M(\theta) = p_\theta(\theta) \cdot \prod_{i=1}^{n} p_{x_i|\theta}(x_i|\theta) \tag{2.40}$$

Note that in order to calculate the MAP estimate, we need to know (or assume) a prior distribution on our parameter θ. A cost function can be applied in the MAP estimate, analogous to the cost function in the hypothesis testing method, that assigns a penalty as a function of $\hat{\theta}$, our parameter estimate and θ, our true, yet unknown parameter. A common cost function is $C(\hat{\theta}, \theta) = |\hat{\theta} - \theta|^2$, and the resulting estimator which minimizes $E[C(\hat{\theta}, \theta)]$ is called the **least square estimate** or LSE. Again, more detailed mathematic description of parametric estimation can be found in chapter 7 of [95].

In the remainder of the thesis, our image segmentation problem will focus almost exclusively on a supervised classification algorithm where the shape of the PDF is assumed to be a multimodal Gaussian shape. The classification component of our solution implements the hypothesis testing method described above. And the PDF

estimation component of our solution implements the EM algorithm described in Section 2.2.2.

Chapter 3

Segmentation Parameters

In the previous chapter we gave some examples of general purpose classification algorithms. In fact, the classification algorithm is one stage of a broader class of problems in pattern recognition. A block diagram showing the basic "pattern" or template of a general pattern-recognition system is shown in Figure 3-1.

This chapter will explore in detail the feature transformation and classification algorithm stages for our specific problem. Recall that the objective of our research is to find a novel approach to segment images or image sequences into regions which correspond to objects. This organization of the image data should be done in a way that is useful for both representation applications (such as compression) and segmentation applications (such as object-based editing). By extracting or "filtering" relevant characteristics from an observed image sequence we are effectively transforming the observation data into useful image features or attributes.

Some of the major challenges with regard to the image attributes addressed in

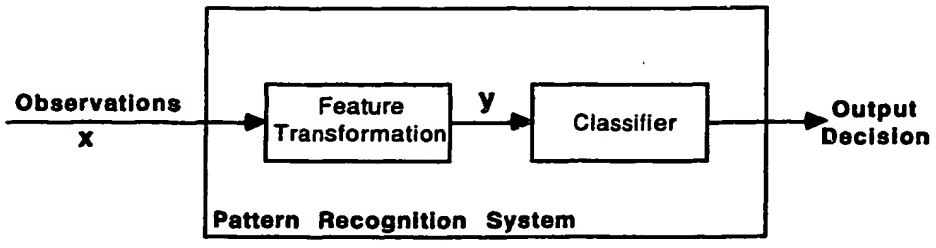


Figure 3-1: Basic template of a pattern recognition system.

this chapter involve: deciding qualitatively which image attributes are most useful, figuring out how to quantitatively calculate those image attributes, engineering an algorithm that analyzes the multiple (potentially conflicting) attributes, and concluding with one classification decision. *I.e.* we need to find a way to "compare apples and oranges".

3.1 Statistical Image Segmentation Framework

Figure 3-2 shows the overall block diagram of the segmentation process. The entire process can be separated into four separate stages. The first stage deals with acquiring and calculating the observation of feature data. The second stage deals with acquiring the training data and tracking or predicting the location of the training data at future frames. The third stage deals with estimating the probability density function of each of the features. Finally, the fourth stage deals with the actual segmentation/decision rule used to classify all the data points into the most likely class.

The objective of our design is to label all the pixels into regions corresponding to "real world" objects or groups of objects. We will measure the performance or success of our segmentation algorithm in Chapter 5 by comparing the results we get from our automated process to a manual segmentation. Chapter 6 will describe how our tool can be used for tasks such as compression, object-oriented editing, and surveillance, among others.

There are several novel concepts in our segmentation design. First, in our approach we wish to be able to segment a broad range of input image sequences with no *a priori* model describing any of the objects or regions we expect to find. Instead, our approach uses a "supervised" segmentation process whereby a user-interface allows the user to provide some labeled training data. A model describing the image statistics of each region is calculated on the fly based on the user training. In contrast to our approach, many existing segmentation schemes are designed with an application in mind that would be useful for finding a specific object or solving a specific task. As a result, the design is extremely restrictive on the types of input images it can

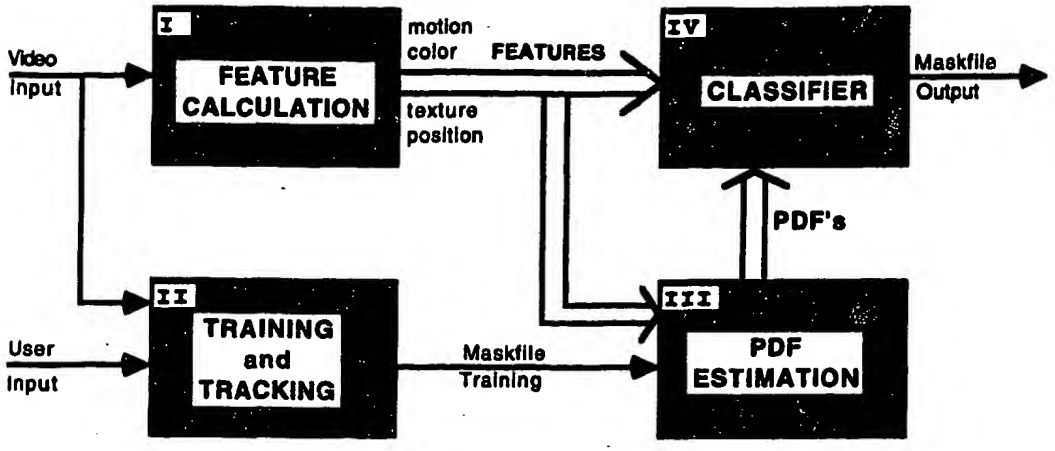


Figure 3-2: General block diagram of segmentation algorithm.

process. These approaches generally have a well developed pre-existing model based on a large accumulation of training data, and are often referred to as a **model driven** system. The main process in a model driven algorithm is generally based on some kind of template matching, either in the image space or some kind of feature space (*e.g.* characteristic texture features or Hough Transform). The input data (or some portions of the input data) is compared to the template/model and decision results are based on the similarity between the data and the model. However, the model driven approach has a much more limited application than our segmentation design and will only work on the task for which it is designed. For example, face recognition or handwriting recognition systems will only detect the features they have been built to recognize. It does not make sense to expect a face recognition system to work on a scanned image of written text, nor does it make sense to expect a handwriting recognition system to work on an image of a person's face. This is because each recognition system works based on a finely-tuned prior model regarding the type of input it expects. In our segmentation design we would like to be able to handle a wide range of input images or image sequences.

A second advantage in our approach is that the object definitions are driven by user training data. As a result, different users can obtain different segmentation results by providing training data accordingly. Typically, approaches that are designed to work on a general class of images use an unsupervised segmentation scheme. The separation

of objects depends on how correlated the data is to itself, and as a result it is difficult to provide control to the user to allow for different segmentation outputs. These unsupervised schemes are also referred to as **data driven** segmentation systems.

A third advantage in our approach is that our decision criterion uses multiple image descriptors including color, motion, texture, and position. By combining multiple cues simultaneously, the segmentation generally performs better than when only one feature is used, such as motion or texture. Also, by using multiple features simultaneously, the exact method of calculating each of the color or motion or texture attributes is not as critical to the segmentation output as it would be if only one attribute were used.

Another novelty in our segmentation framework is that every sample in the image sequence is classified. Other segmentation approaches label an entire block as belonging to a particular region In a block-based classification scheme, each image in an image sequence or data base is arbitrarily broken up into tiles or blocks. A set of features is used to characterize each tile, and these features serve as a metric that determines the proximity or similarity between the blocks. The supervision or training comes from a user who labels sample blocks and the algorithm finds the best matching blocks in the remaining image sequence or data base. As a result of this paradigm, only blocks are labeled and it is difficult to extract regions which correspond to "real world" objects, which is one of the goals of our segmentation design.

Finally, by using a more robust PDF estimation technique (modeling the distribution of pixel membership as a multimodal Gaussian curve instead of a simple unimodal Gaussian) we are able to describe more complicated behaviors of the image characteristics. For example, if one object is spatially disjoint and has components on both the left and right side of the image, then by using a multimodal distribution model to estimate the likelihood of the position of the object we get a much more accurate estimate than one with likelihood modeled by a unimodal Gaussian curve. As an example, refer to the top image of Figure 6-1(b) and notice that a "plants" object lies on both the bottom right and left corner of the image. The multimodal modeling benefit holds true for all of the image features, which means we can more

accurately handle heterogeneous objects.

We mentioned earlier that our segmentation algorithm has the ability to produce different segmentations based on two different sets of training data, and that it is difficult for an unsupervised segmentation to do the same. Let us take a closer look at segmentation approaches that attempt to label or cluster groups of pixels completely autonomously. Typically, a statistical decision criterion such as ISODATA is used. Image characteristics are calculated from the observation data, or the image itself is used as the feature space. Pixels are initially assigned at random (or quasi-randomly) to one of R classes, where R is an initial estimate of the total number of regions. The mean and variance of each cluster is computed, clusters with close means are merged, and any cluster with a large variance is split into two clusters. At each iteration, points are individually reassigned to the most "similar" region, thereby shifting the mean and variance. The split and merge decision, as well as the reclassification decision, is iteratively recomputed until there is little or no change in classification between iterations.

However, this class of unsupervised segmentation approaches generally runs into a fundamental problem related to scale or the number of regions in an image. Specifically, the resulting segmentation depends critically on the choice of parameter, R, and it is not always clear what value this parameter should take on. Objects or regions in natural images are often composed of multiple and typically differing components. Thus it is very difficult (even for people!) to convene on: a) **what** the different objects or regions are in the image; and even b) **how many** different objects exist in the image. For example, a city building can be considered as one object, or all of the windows from the building can be considered as one region (and the rest of the building a second region), or perhaps each window might be considered to be separate objects, etc. The desired segmentation is therefore a function of the user or specific task at hand. Moreover, it is difficult for an automatic image segmentation process to succeed without a pre-existing model or template, or without some kind of user-supplied points.

Our approach is quite unique in that we use a supervised segmentation algorithm,

yet we do not use pre-existing models or templates of specific objects. Instead, our approach is more statistical in nature and is in that respect similar to the unsupervised scheme. Training points are supplied by the user who highlights representative points from each of the user-defined regions. Thus a statistical model of each region can be created on-the-fly for each region from the user-supplied training points. The remaining unlabeled pixels are assigned to the region with highest statistical similarity. Using this approach, we can successfully form several varying segmentations of the same image sequence, depending on the user's needs. Since there is a very high correlation between the user's training points and the underlying regions these training points represent, it is simple enough to steer the algorithm to a different segmentation by simply changing the training samples accordingly.

Our approach seems to have the characteristics of both a model driven segmentation procedure and one that is data-driven, since there is no fixed model or set of models typically found in template-matching segmentation algorithms, yet the segmentation is not produced entirely autonomously either. Technically, ours is a model driven approach, where the model varies adaptively based upon the user training. The model is statistical in nature because we analyze the distribution of the image attributes for each region of the user training. By estimating multiple models simultaneously (one for each region), we are not limited to recognizing one object exclusively (which is typical of model-driven systems) but rather we have developed a general tool that can be used for a broad variety of tasks.

3.2 Image Attributes

Much of the research in image processing and image understanding has focused on extracting qualitative information from an image or image sequence using quantitative methods. These qualitative attributes tend to capture image properties such as color, texture, motion, shape, position (both local and global) and depth, among others. In our segmentation process we use a combination of many of these image attributes. However, we do not incorporate knowledge-based attributes. For example, we are

not trying to extract an information metric that assimilates an image of the Empire State Building with an image of the Statue of Liberty because they are both famous landmarks associated with New York City.

One major advantage of using multiple features simultaneously to produce one decision is that the burden of having the "perfectly calculated" feature is diminished. Specifically, if the features were used individually, there would be a greater need to ensure that the optimal feature calculation is used. In actuality, the features tend to serve more as a qualitative estimate than an exact measurement. Consequently, sometimes "Method A" of estimating a feature serves better than "Method B" of estimating the same feature, and other times, vice versa. The motion estimation feature, for example, is one that is known to be an ill-posed problem. Namely, there is no guarantee that the estimation will be a precise measurement at every single pixel, but rather it serves as more of a qualitative measurement, over a larger area. A statistical classifier using only motion information will generally not perform as well as one which uses multiple features, regardless of the accuracy (within practical limits) of the motion estimator.

3.2.1 Motion Estimation

There are several methods of estimating the motion in an image, including correlation methods such as block matching, constant brightness constraint methods such as optical flow, and rigid motion body constraints such as affine motion estimation. In this section we will take a look at these three methods of motion estimation. In our segmentation experiments, however, we use two different methods of computing optical flow as our motion attribute.

We have found that our segmentation results do not depend on which motion estimation technique is used, even though there may be a noticeable difference at the estimation stage. One major reason why there is not a great difference between the segmentation results is that motion is only one of our image attributes. Thus, changing the computation of only one of the attributes while keeping the other attributes unchanged does not considerably change the segmentation result. This shows that

the segmentation process is actually quite stable, as minor changes in the input stage do not seriously affect the output stage. A second reason for this stability effect is because in segmentation we are not concerned with the exact motion calculation of each pixel, but rather we are more concerned with the similarity between the motion estimation at each pixel and the PDF estimation calculated from the training points. So to determine the effectiveness of a motion estimation technique when applied to segmentation, we are interested in how well the difference in motion of distinct regions is captured. The difference in motion between regions could be captured by different motion estimation techniques independent of the fact that there might even be a major difference between these motion estimation techniques.

Block Matching

The first motion estimation method we discuss is one based on searching, where a block in one frame correlates most with a block in the next (or previous) frame. Such a technique is known as **block matching**. At each pixel location $\{x, y\}$ we want to find the motion vector $\{v_x, v_y\}$ over a predetermined search range S such that the error (or difference) $E(v_x, v_y)$ between two successive frames in an image sequence is minimized within a specified block region R.

$$E(v_x, v_y) = \sum_{(x,y)\epsilon R} (I(x + v_x, y + v_y, t_1) - I(x, y, t_2))^2 \tag{3.1}$$

$I(x, y, t1)$ and $I(x, y, t2)$ represent the luminance at frames t_1 and t_2 where $t_1 < t_2$. The $\{v_x, v_y\}$ pair, which corresponds to a minimum error or difference $E(v_x, v_y)$, represents our block-matching motion vector estimate at the position x, y. Note that the search range extends in the t_1 frame, and that the position at frame t_2 is kept constant. This search is often called a **reverse** block matching algorithm (as opposed to a **forward** one) because for each block in frame t_2 we search the previous frame t_1 for a block with the highest correlation.

The summation is two-dimensional over a block range R. Typically the block size is anywhere from 8x8 to 16x16 pixels, though the horizontal and vertical dimensions

of the block size are not required to be the same size. The v_x and v_y parameters vary over a search range S. The number of pixels in the search range generally depends on how much motion is in the sequence, as well as the spatial size (number of pixels per row and column) of the image sequence. Typically, S stretches from about -12 to $+12$ pixels to as high a range as -30 to 30, for each of v_x and v_y. Again, the horizontal and vertical parameters are not required to have the same value, and for many sequences it makes sense for the horizontal search range to be larger than the vertical one.

If non-overlapping blocks are used, the block matching algorithm results in a coarse motion estimation. Overlapping blocks are used to generate a motion estimation vector at each pixel. While this second method is much more computationally intensive, it is necessary for our segmentation process, since all pixels must be labeled in the end. Overall, block matching algorithms are quite useful because they are not too complex to be practically implemented, and because they provide a good "statistical" estimate of each pixel's motion, on average providing a close approximation to each pixel's true motion. The main disadvantage is that the underlying motion model assumed in block matching is too simple. Specifically, the method can only model translational motion, which is not always typical of natural motion. A secondary disadvantage is that there is no coherency stipulation between blocks. Thus the motion estimation between neighboring blocks have no effect on each other, when in reality these two neighboring blocks may actually be a part of the same object or region.

Optical Flow

An alternative method to block matching is called **optical flow**. Introduced by Horn and Schunck [64], optical flow estimates the apparent motion of regions between two successive frames in an image sequence by measuring the change in the brightness pattern between those frames. Although there are many methods for measuring these changes in brightness and how these changes directly or indirectly affect the estimated motion, in this section we will focus primarily on two basic approaches. One approach, introduced by Lucas and Kanade [69], uses a local smoothing constraint, while the

other approach by Horn and Schunck [64] uses a global smoothing constraint.

Both approaches assume that changes in intensity over time are due mostly to motion. Thus we can say:

$$f(x, y, t) \approx f(x + dx, y + dy, t + dt) \tag{3.2}$$

where $f()$ represents the image intensity, and dx, dy, and dt represent small changes in position x,y and time t respectively. Though the optical flow motion estimate is not exactly equal to the true motion field, it is a good approximation when the first order Taylor series approximation of $f(x, y, t)$ holds:

$$f(x + dx, y + dy, t + dt) = f(x, y, t) + \frac{\partial f}{\partial x}dx + \frac{\partial f}{\partial y}dy + \frac{\partial f}{\partial t}dt. \tag{3.3}$$

By subtracting both sides of the equation from $f(x, y, t)$, dividing by dt, and using the approximation in Equation 3.2 we get:

$$\frac{\partial f}{\partial x}u + \frac{\partial f}{\partial y}v + \frac{\partial f}{\partial t} \approx 0 \tag{3.4}$$

where $(u, v) = (\frac{dx}{dt}\frac{dx}{dt})$ is the optical flow motion estimation vector, and gets calculated at each pixel x, y, t in the image sequence. Equation 3.4 is known as the **Brightness Change Constraint Equation** or BCCE.

The Lucas and Kanade solution for the optical flow vector implements a least squares error (LSE) calculation over a local region. First we rewrite the optical flow equation as $-f_t \approx f_x u + f_y v$ where (u, v) is the optical flow vector, and f_x, f_y, and f_t are the partial derivatives with respect to x, y, and t. Next, we want to minimize the mean square error with respect to (u, v) over a local region R, where the error e is defined by:

$$e = f_t + f_x u + f_y v \tag{3.5}$$

By taking the derivative of the local mean square error with respect to u and v, we

get:

$$\frac{\partial}{\partial u}\sum_R e^2 = 2\sum_R e f_x = 0 \tag{3.6}$$

$$\frac{\partial}{\partial v}\sum_R e^2 = 2\sum_R e f_y = 0 \tag{3.7}$$

$$\tag{3.8}$$

This pair of equations can be expressed in matrix form as:

$$\begin{bmatrix} \Sigma f_x^2 & \Sigma f_x f_y \\ \Sigma f_x f_y & \Sigma f_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \Sigma f_t f_x \\ \Sigma f_t f_y \end{bmatrix} \tag{3.9}$$

Notice that with this LSE approach, the under-constrained problem prescribed by Equation 3.4, which has two unknowns to solve for and one equation, is converted to an over-constrained set of equations. Recall that we can model an over-constrained system as $\mathbf{A}\bar{x} = \bar{b}$, where $\mathbf{A}$ is a matrix, $\bar{x}$ is a vector of unknowns, and $\bar{b}$ is a (known) vector. (Note that $\bar{x}$ does not mean the average value of x in this case.) The number of rows and columns in $\mathbf{A}$ specifies the number of equations and unknowns, respectively, in the system. In our example, $\mathbf{A}$ has two columns, since our unknown vector $\bar{x}$ has two elements (u, v). The number of rows in $\mathbf{A}$ is equivalent to the size of the local region R over which we minimize the error. Each row of $\mathbf{A}$ contains two elements (f_x, f_y) evaluated at different positions of x and y in our local region R. Similarly, the $\bar{b}$ vector contains values of $(-f_t)$ at different positions. Equation 3.9 corresponds to what are known as the "normal equations," namely:

$$\mathbf{A}^T\mathbf{A}\bar{x} = \mathbf{A}^T\bar{b} \tag{3.10}$$

and of course the solution to the normal equations is

$$\bar{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\bar{b} \tag{3.11}$$

The Lucas and Kanade method described above uses a local constraint to solve

for Equation 3.4. Another approach, introduced by Horn and Schunck, uses a global constraint: that the motion vector (u, v) is relatively smooth between neighboring pixels. Thus, a second constraint is to choose a solution that minimizes the magnitude of the gradient of the optical flow. Namely:

$$e^2_{smooth} = u_x^2 + u_y^2 + v_x^2 + v_y^2 \tag{3.12}$$

The complete solution is to find (u, v) that minimizes a combined error function:

$$e^2_{total} = \sum e^2_{bcce} + \alpha e^2_{smooth} \tag{3.13}$$

where e_{smooth} is defined as Equation 3.12, e_{bcce} corresponds to the error term defined in Equation 3.5, and α is a weighting factor that determines the relative importance of the two constraints.

The full details of the solution to these equations can be found in [64]. Stated briefly, it involves an iterative approach, where at each iteration the (u, v) estimate is a function of a local average of the previous iteration, plus or minus a correction term which is a function of the e_{bcce} variable. Because of the global smoothing constraint, there is an implicit image intensity model which likens the image to a "rubber sheet" that distorts over time.

In general, optical flow motion estimation tends to give a better estimate to the true motion of regions than block-matching estimates. However, there are problems in the optical flow estimates as well. First, the underlying assumptions of the BCCE equation does not take into account areas in the image where occlusion occurs (i.e. it assumes that regions do not disappear from view). Second, because the problem is under-constrained, there is no guarantee that a unique solution exists, as in the case of an aperture problem, or areas of flat surfaces. Third, even though the motion field is dense, the resulting motion estimate is simple translational motion at each pixel. The sense that each pixel belongs to a larger object or region is still absent.

Affine Parameter

An **affine** motion model is more general than a translational model as it allows for the representation of more complicated motions such as rotation, zooming, shears, and translations, as well as linear combinations of these movements. Given an estimate of a dense motion field (u, v) we can use a 2-dimensional affine model to solve for the parameter vector (a, b, c, d, e, f) which approximates the motion vector as per the equations:

$$u(x, y) = a + bx + cy \tag{3.14}$$

$$v(x, y) = d + ex + fy \tag{3.15}$$

A LSE method is utilized to solve these two equations with a total of six unknown parameters (a, b, c, d, e, f). The affine motion model parameters can be solved for in one of three manners: A) A sparse block-based approach, B) A dense block-based approach, or C) A region-based approach.

In method A, the dense optical flow velocity vector values within a block are used to determine the best (in a least squares sense) affine parameter vector for that entire block. Thus if one frame of the image is 512 by 512 pixels, and the region block size is 16 by 16 pixels, a total of 32 by 32 affine parameter vectors are found; this is referred to as a sparse affine parameter model. In method B, the approach is similar to method A, except overlapping block regions are used in order to get a dense affine parameter estimation. Method C differs in that the values of the optical flow estimate vector used for the LSE solution are not drawn from a block or regular pattern, but are instead taken from a region. However, since segmentation is our desired goal, given the image attributes we find ourselves in a boot-strap problem if the image attribute calculation depends on the segmentation result.

Because of this dependency complexity, the motion feature used for our segmentation experiments was calculated using the optical flow calculation described in Section 3.2.1. The reason why the affine parameter motion estimation method was not used as per methods A and B, is because the first, method A, does not result in

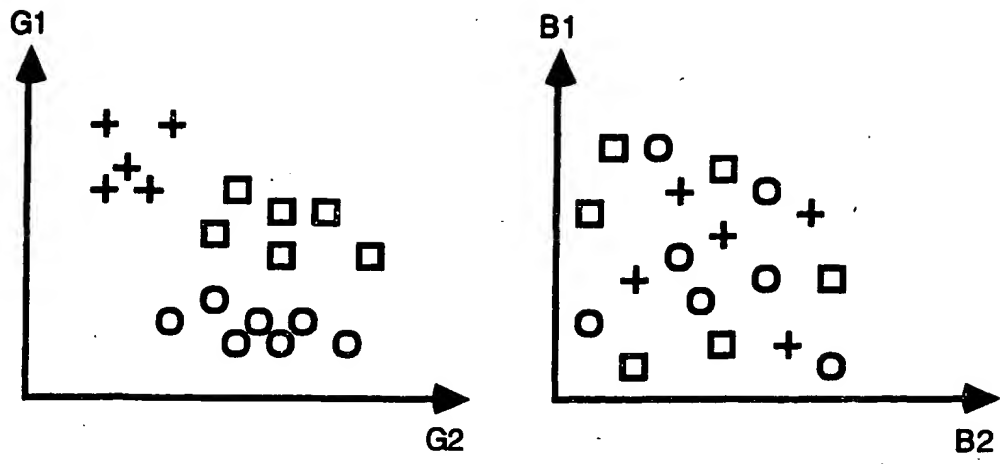


Figure 3-3: Example two-dimensional feature space, (G_1, G_2), producing a desirable partition vs. two-dimensional feature space, (B_1, B_2), producing an undesirable partition.

a dense affine parameter estimation, which is necessary for our framework in which each pixel must be labeled. The main problem with affine estimation using method B is that the estimation is too sensitive to the choice of block size.

3.2.2 Texture

In much the same way that there are many techniques to estimate motion throughout an image sequence, there are also many different methods to compute texture information from an image. We are interested in calculating texture information or features that will be used for segmentation and not for representation. These texture analysis methods generally fall into two categories: structural methods or statistical methods. In our experiments we are not quite so interested in finding primitive components which describe structural patterns, since these are typically useful for representing objects with a regular geometrical pattern (such as bricks, wire fences, reptile skin, etc.). Rather, we are looking for texture variations that are better described by statistical models. This section will focus on these statistical models.

Recall that in our pattern recognition (or image segmentation) framework we use training samples from our feature vector to partition the feature space into regions which correspond to training classes. Therefore, we need to calculate attributes from

the image which serve as distinguishing characteristics. Figure 3-3 shows a two-dimensional feature space which corresponds to features that produce a desired space partition.

Although texture measurements exist that which are based upon motion, in our experiments, we focused only on intra-frame texture parameters. One main reason why we decided to restrict the computation of texture to an intra-frame paradigm is that we combine motion information with the other features (including texture) in our segmentation stage, so there is no need to do the combination at the feature calculation stage. A second reason is that while there are some texture estimates which use motion, most of the traditional texture measurements are intra-frame, thereby allowing us to have a greater number of methods of texture computation to choose from.

Below we describe three methods of computing texture information. The first method, which is called **local statistics** is a very simple texture calculation which estimates a local mean and variance at each pixel. The second, called **simultaneous autoregressive model** or **SAR**, uses a linear prediction method to try to estimate the current pixel as a function of its neighbors. The weighting coefficients of each of the neighbors, along with an error term, serve as the texture features. The third method, called **frequency analysis**, uses a combination of horizontal, vertical, and diagonal filters, or a Fourier transform at multiple scales of the image.

Local Statistics

When we first started to implement our segmentation algorithms, we used motion information only. When we added texture information, the simplest texture calculations were local statistics on the image intensity channel. Specifically, at each point in the image (x, y) we compute the local mean $M(x, y)$ and the local standard deviation $S(x, y)$ of the image intensity $I(x, y)$ in a local region R. In the equations below, W

represents the size of the region R.

$$M(x,y) = \frac{1}{W} \sum_{(x,y)\epsilon R} I(x,y) \tag{3.16}$$

$$S(x,y) = \left[\left(\frac{1}{W} \sum_{(x,y)\epsilon R} I^2(x,y) \right) - M^2(x,y) \right]^{1/2} \tag{3.17}$$

For our experiments, the shape of the region was chosen to be a block, and the size varied from 3 to 15 pixels in width and height. The local statistics texture measurement produces two parameters at each point in the image.

Since the result of the texture measurement depends on our choice of block size, and as we don't know in advance what the best choice of block size should be, we decided to use the local statistics at a multiscale level. In the multiscale version we compute multiple estimates of the mean and standard deviation using multiple values of the window block size. Initially we experimented with two values of window size, producing a total of four texture features: two for the mean and two for the standard deviation at each block size level. We then tried the multiscale method using three separate values of the window size, producing six texture features. However, two levels presented too much of a spread between block sizes, and three levels produced too many texture parameters. Finally, we tried a cross between the multiscale statistics and the uni-scale method. In particular, the mean was calculated at three block size levels while the standard deviation was calculated only at the middle block size level. Typical values were three-, nine- and fifteen-pixel blocks for the mean, and a block size of nine pixels for the standard deviation feature.

The main advantage of the local statistics texture feature is that it is a very quick and easy way to get an approximation of the texture characteristics. Figure 3-4 shows how our two-feature local statistics estimates would serve as good features for a statistical segmentation task. In our example, we show how the local mean and local standard deviation texture calculation serve as a desirable choice of two features and how they could represent six texture classes.

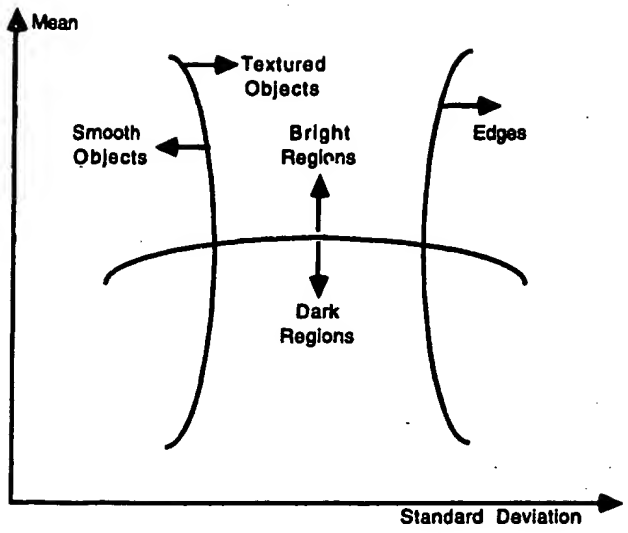


Figure 3-4: Possible texture class partitions from a local mean and standard deviation feature space.

Simultaneous Auto Regressive Model

The basic Simultaneous Autoregressive (SAR) model for texture classification and segmentation is described by Mao and Jain in [66], along with variations of the basic model. SAR is a statistical method which assumes texture can be characterized by measuring spatial interactions among neighboring pixels. Specifically, the texture of a region can be understood by first expressing each pixel as a weighted sum of its neighbors and then analyzing the respective weights. This idea is consistent with the notion that texture is essentially a description of the difference or similarity between neighboring pixels.

In our SAR model, we use the the weighted sum of eight neighboring pixels to represent a given pixel. The process is repeated for every pixel in an NxN region R with the intent of averaging the weights gathered at each pixel to create the best possible weights of the immediate neighbors of the pixel at the center of the region. Let $I(x, y)$ represent the image intensity at every point in the image. Let us define $f_0(i, j)$ to represent the intensity at each pixel location $(i, j) \epsilon R$, where R is defined by an NxN block within the image that is centered around the pixel located at $I(x, y)$. By moving this 3x3 window throughout the NxN block, we can compute the weights

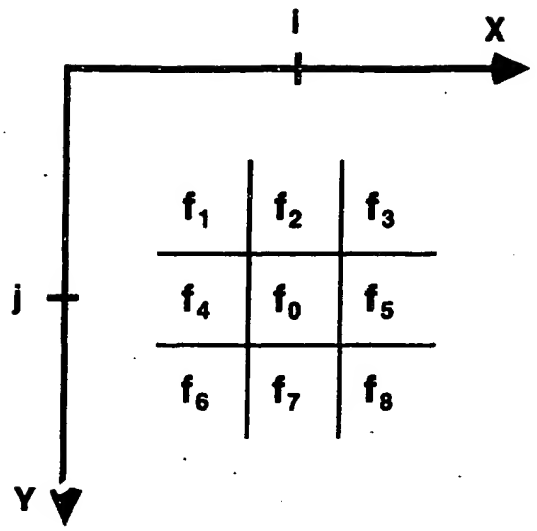


Figure 3-5: Neighbor labels for a one level SAR texture model at one sample point in the image ($x = i, y = j$). f_1 to f_8 are the neighbors centered around point f_0.

$c(x,y)$ and error term $e(x,y)$ which form the best solution to the equation:

$$\begin{aligned} I(x,y) &= c_1 I(x-1,y-1) + c_2 I(x,y-1) + c_3 I(x+1,y-1) + c_4 I(x-1,y) \\ &\quad + c_5 I(x+1,y) + c_6 I(x-1,y+1) + c_7 I(x,y) + c_8 I(x,y+1) + e(x,y) \end{aligned} \tag{3.18}$$

We define $f_k(i,j)$, for $1 \leq k \leq 8$, as the eight neighboring pixels of $f_0(i,j)$. The index k moves in raster scan order from the top left neighbor of pixel (i,j) to its neighbor on the lower right (see Figure 3-5).

$$\begin{aligned} f_0(i,j) &= I(i,j) \\ f_1(i,j) &= I(i-1,j-1) \\ f_2(i,j) &= I(i-1,j) \\ f_3(i,j) &= I(i-1,j+1) \\ f_4(i,j) &= I(i,j-1) \\ f_5(i,j) &= I(i,j+1) \\ f_6(i,j) &= I(i+1,j-1) \end{aligned}$$

$$f_7(i,j) = I(i+1,j)$$
$$f_8(i,j) = I(i+1,j+1)$$

Note that when the center pixel of our 3x3 neighborhood is the center pixel of our NxN region, $f_0(i,j) = I(x,j)$, which is the value of the pixel we wish to represent as a function of its neighbors. We use the following equation to model our estimate $f_0(i,j)$

$$f_0(i,j) = \sum_{k=1}^{8} c_k(i,j) f_k(i,j) + e(i,j) \tag{3.20}$$

The $c_k(i,j)$ parameters are our coefficients for each of the neighbors, where (i,j) represent the pixel location for which we are computing the texture parameters, and $e(i,j)$ represents the error or offset of an estimate from just the neighbors. For symmetry reasons, we assume *a priori* that the c_k and c_{9-k} coefficients (corresponding to the f_k and f_{9-k} values) will equally contribute to our estimate of f_0 (i.e. the left neighbor of a pixel contributes as a much as the right neighbor). Thus we can reduce the eight coefficients $c_k(x,y)$ to four new coefficients $a_k(x,y)$. We also for convenience define $g_k(i,j) = f_k(i,j) + f_{9-k}(i,j)$ for $1 \leq k \leq 4$. Thus our model from Equation 3.20 can be rewritten as

$$f_0(i,j) = \sum_{k=1}^{4} a_k(x,y) g_k(i,j) + e(x,y) \tag{3.21}$$

This still leaves us with five unknowns, and only one equation. A common solution is to find an LSE estimate in much the same way we solved for the optical flow coefficients described in Section 3.2.1. The LSE solution for coefficients $a_k(x,y)$ and error $e(x,y)$ is calculated over the NxN region centered around the point (x,y). Thus we would evaluate Equation 3.21 at N^2 different values of (i,j) resulting in N^2 equations and five unknowns ($a_1(x,y),\ldots,a_4(x,y)$, and $e(x,y)$). These five values serve as our texture features or attributes. Since we need to compute these attributes at each sample in the image, we simply move the NxN region over which we apply the least squares error estimate. Suppose our LSE region is 15x15 pixels and our image is 512x512 pixels. To compute the five texture parameters at $(x,y) = (27,61)$

for example, we can generate the following matrix equation:

$$\overbrace{\begin{bmatrix} g_1(20,54) & g_2(20,54) & g_3(20,54) & g_4(20,54) & 1 \\ g_1(21,54) & g_2(21,54) & g_3(21,54) & g_4(21,54) & 1 \\ & & \vdots & & \\ g_1(34,54) & g_2(34,54) & g_3(34,54) & g_4(34,54) & 1 \\ g_1(20,55) & g_2(20,55) & g_3(20,55) & g_4(20,55) & 1 \\ g_1(21,55) & g_2(21,55) & g_3(21,55) & g_4(21,55) & 1 \\ & & \vdots & & \\ & & \vdots & & \\ g_1(34,68) & g_2(34,68) & g_3(34,68) & g_4(34,68) & 1 \end{bmatrix}}^{\mathbf{A}} \overbrace{\begin{bmatrix} a_1(x,y) \\ a_2(x,y) \\ a_3(x,y) \\ a_4(x,y) \\ e(x,y) \end{bmatrix}}^{\vec{x}} = \overbrace{\begin{bmatrix} f(20,54) \\ f(21,54) \\ \vdots \\ f(34,54) \\ f(20,55) \\ f(21,55) \\ \vdots \\ \vdots \\ f(34,68) \end{bmatrix}}^{\vec{b}} \tag{3.22}$$

We solve this in the same way we solved Equation 3.10.

Loosely speaking, the components show the direction of the general orientation, on average, for each NxN region. Specifically, if the a_1 coefficient contributes most, the general orientation is diagonal (up-left to down-right), if a_2 contributes most, the orientation is vertical, if a_3, the orientation is diagonal (up-right to down-left), and a_4 corresponds to a horizontal orientation. The e term represents how much error or variance there is in our estimate of the center pixel using just the four coefficients.

The SAR texture feature calculation depends on the choice of how large we choose our NxN region over which we find the LSE, and also depends on the choice of neighbors and size of the neighborhood over which we do our current pixel prediction. Two variations to the standard SAR model are called Multiresolution SAR (MSAR) and Rotational Invariance SAR (RISAR).

In the MSAR method, the neighborhood region is done at multiscales, and at each scale the five texture features are computed. Specifically, at each computation level p of the texture coefficients, Equation 3.20 is modified so that the $i \pm 1$ and $j \pm 1$ indices become $i \pm p$ and $j \pm p$, so that at each level you represent a given pixel as a function of increasingly distant neighbors. In the RISAR method, the shape of the neighborhood region is circular, instead of square. As a result, the diagonal neighbors are often not

aligned exactly with the image pixel grid. A bilinear interpolation scheme is used to estimate the value of these off-axis neighbors, based on the four nearest on-axis points.

For the purpose of our experiments, the SAR texture features were calculated using the standard SAR model. This was mostly due to the fact that standard SAR is straightforward to calculate, the MRSAR model produces too many parameters, and the RISAR did not produce a much different result than standard SAR.

Spectral Frequency Analysis

The final statistical texture feature estimate we will discuss is known as spatial frequency analysis, and is also referred to as steerable filters, oriented filters, or directional derivatives. We will not get into too much specifics of how these textures are calculated; we instead refer the reader to [15], and [7].

The basic calculation for most of these spatial frequency texture methods is that the energy in the filtered image is treated as the feature. The difference between some of these methods is in how the image filter space is partitioned, the directionality of the filters, and whether or not a multiscale method is used. For example, some methods use a polar coordinate system and partition the frequency space into different classes of frequency magnitude and angle for each feature. Other methods use a rectangular coordinate system along with a set of oriented filters. The different feature values would come from the horizontal, vertical and diagonal directions, and for each of these directions there could be both a low pass and high pass filter applied. Both methods could also be applied in a multiscale manner.

In our segmentation experiments discussion, we only implement the local statistics method and the SAR method of computing texture features described in Sections 3.2.2, and 3.2.2, respectively. We omit the oriented filters method because experimentally we found that the number of dimensions in the feature space which resulted from using this method was too high. A second reason is because the calculation does not lend itself well to calculating texture parameters at every point in the image due to its multiscale nature. Third, the local statistics method is actually

a special case of the oriented filters method, with the low pass and high pass filters corresponding to the local mean and standard deviation calculations. Finally, preliminary tests showed that the final segmentation results did not vary much depending on which texture calculation was used. Again, as with the motion features, the crucial step is to include all the attributes (texture, motion, and color) in the segmentation stage, and the method used to calculate each of these attributes is not critical. For some images, one feature calculation method will yield a slightly better segmentation, and for other images another method will yield a slightly better segmentation result.

3.2.3 Color and Position

Little or no calculation is required for both the color feature and the position feature. Recall that a "feature" is a transformation of observed data. Since in this case our observed data is a color image sequence, we already have the color information at each pixel. Likewise, we already have the position information at each pixel, since it is a two-dimensional feature containing the horizontal and vertical image index of each pixel location.

Color Spaces

Color space is three dimensional, and a single color value can be represented by any one of many color coordinate systems. We will focus on three primary color spaces, namely **RGB**, **YIQ**, and **LAB**. Most computer displays use the RGB format, corresponding to the red, green, and blue value of each pixel. The raw image sequence data is also generally stored in RGB format. The YIQ format evolved from the National Television Standards Committee (NTSC). Specifically, the color television system was to be backward compatible with the existing black and white system, so that people with black and white television sets could view color broadcasts. Thus, one of three dimensions of the color space was constrained to be the luminance, or Y signal. The other two components were selected as the in-phase (or I component) and the quadrature-phase (or Q component). The I component falls closely on the

red-cyan axis and the Q component on the green-magenta axis.

To convert from RGB to YIQ (and vice versa) there is a pair of simple matrix conversion equations. Equations 3.23 and 3.24 show the conversion matrices from RGB to YIQ and YIQ to RGB respectively.

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.2986 & 0.5872 & 0.1142 \\ 0.5953 & -0.2737 & -0.3216 \\ 0.2108 & -0.5219 & 0.3111 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{3.23}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.0000 & 0.9577 & 0.6229 \\ 1.0000 & -0.2721 & -0.6483 \\ 1.0000 & -1.1052 & 1.7046 \end{bmatrix} \begin{bmatrix} Y \\ I \\ Q \end{bmatrix} \tag{3.24}$$

There is often quite a bit of variation between displays, even when they are displaying the same RGB or YIQ values. To offset this potential problem, in 1931 the CIE (Commission International de l'Eclairage) developed a color space based on the notion of a "standard observer" [55], and defined a linear transformation such that the color of an object can be specified by a set of values (X, Y, Z) known as the **tristimulus** values. Although colors are hardly ever referred to by their XYZ values, they do have the advantage that the perceived color in XYZ space is device independent.

To convert the RGB values of a monitor to the CIE standard of XYZ tristimulus values, usually a factory-aligned specification standard (e.g. D6500) for the monitor is known. The standard has reference chromaticity values (easily calculated from the tristimulus values) for each of red, green, blue, and white. Based on these reference values, a conversion matrix M, shown in Equation 3.25 can be calculated to convert from RGB to CIE-XYZ. The details of how to solve for the matrix M are found in Appendix A.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \overbrace{\begin{bmatrix} 0.4782 & 0.2986 & 0.1746 \\ 0.2635 & 0.6550 & 0.0815 \\ 0.0198 & 0.1604 & 0.9079 \end{bmatrix}}^{M} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{3.25}$$

While the use of the CIE-XYZ color space allows for colors to be device independent, there is yet another problem: the color space is sampled in a highly non-uniform manner. In an attempt to solve this problem, in 1976 the CIE introduced two uniform color spaces, one of which is known as CIE-LAB [55] or just LAB. Once the values are converted from RGB to the CIE-XYZ standard, a second step which applies a non-linear transformation is required to convert each pixel value from CIE-XYZ space to LAB space. The L values represent luminance and the A and B values are the red/blue and yellow/blue chrominance signals. The equations which describe this conversion are:

$$L = \begin{cases} 116(\frac{Y}{Y_n})^{\frac{1}{3}} - 16, & \frac{Y}{Y_n} > 0.008856 \\ 903.3(\frac{Y}{Y_n}), & \frac{Y}{Y_n} \leq 0.008856 \end{cases} \tag{3.26}$$

$$A = 500\left[f\left(\frac{X}{X_n}\right) - f\left(\frac{Y}{Y_n}\right)\right] \tag{3.27}$$

$$B = 200\left[f\left(\frac{Y}{Y_n}\right) - f\left(\frac{Z}{Z_n}\right)\right] \tag{3.28}$$

where the function f is defined as:

$$f(t) = \begin{cases} t^{\frac{1}{3}}, & t > 0.008856 \\ 7.787t + \frac{16}{116}, & t \leq 0.008856 \end{cases} \tag{3.29}$$

where the values X, Y, Z correspond to the tristimulus pixel value, and the X_n, Y_n, and Z_n correspond to the tristimulus values of the reference white color. See Appendix A for a more detailed explanation.

Experimentally, the image data in the LAB and YIQ spaces usually helped to produce a better segmentation, although the difference was generally quite small. As noted in the sections discussing motion and texture, the largest contributions to the segmentation results do not depend critically on exactly how the feature is calculated (*e.g.* RGB, YIQ or LAB) but rather that a collection of features are being used simultaneously.

3.3 Tracking Method

Stage II of Figure 3-2 concerns the estimation of the predicted location of the training points identified in the first frame. This problem is practically identical to the motion estimation problem discussed in Section 3.2.1. The main difference in our tracking algorithm, however, is that we do not need to estimate the motion of every sample and instead track the position of the points corresponding to the training points acquired in the first frame. In tracking the training points from frame to frame, we assume that if a training sample is labeled to a particular class ω_i in the first frame, its classification will not change over time.

Section 3.3.1 discusses a **point-based** tracking scheme where training points belonging to the same region are not constrained to have similar motion. The motion of each training point is calculated independently using a type of block matching algorithm. Section 3.3.2 discusses a **region-based** tracking scheme in which all of the training points belonging to the same region are constrained to fit an affine motion model. Thus all of the training points in one region are used collectively to estimate the one rigid-body motion for that region. Each region is described by a motion estimate in the region-based method, while in the point-based method each point is described by a motion estimate.

3.3.1 Point-Based Tracking

Our point-based method of tracking is very similar to the block matching method described in Section 3.2.1. For our region tracking experiments, we first convert the RGB image to YIQ, as defined by Equation 3.23. (Note: we can alternatively convert the image into YBR space. Y, the luminance signal is the same as the Y in YIQ space, but the color components BR are proportional to $Blue - Y$ and $Red - Y$ and have the advantage that they are restricted to positive values because a gain and offset are applied.) Once the image is converted to YIQ space, we perform the block matching search at the full resolution level on the luminance (Y) channel, but at half the resolution level in the color (IQ) channels. (Half resolution implies the IQ

channels are low-pass-filtered and downsampled by a factor of 2, effectively reducing our search range and block size by a factor of 2.) The result is a faster implementation, using color at half the resolution with almost no loss of quality than when using color at the full resolution.

One major difference between block matching for motion estimation purposes and block matching for tracking purposes is that for tracking there is the possibility of propagation of error. Suppose we define P_0 as one of our reference training points (selected by the user) whose spatio-temporal position is denoted by (x_0, y_0, t_0). Let us further use the notation P_i as our best estimate of where P_0 has moved to in the i^{th} frame. The task therefore is to solve for the location (x_i, y_i, t_i) of P_i at each frame t_i, where $i > 0$. By using the block matching algorithm, we can estimate (x_1, y_1, t_1) by comparing the MSE between a block around P_0 and a series of blocks within a search range neighborhood of (x_0, y_0, t_1). For $i = 1$, $(x_1, y_1) = (x_0 + v_x, y_0 + v_y)$ where (v_x, v_y) is solved as per Equation 3.1. For frames $i > 1$, one could either: A) compare the MSE between a block around P_{i-1} and a series of blocks within a search range neighborhood of (x_{i-1}, y_{i-1}, t_i), B) compare the MSE between a block around P_0 and a series of blocks within a search range neighborhood of (x_{i-1}, y_{i-1}, t_i), or C) compare the MSE between a block around P_0 and a series of blocks within a search range neighborhood of (x_0, y_0, t_i). Note that at $i = 1$ the three comparison methods A,B, and C are identical. Also, if the error in the best match is extremely low (or zero), effective comparison methods are also quite similar. However, let us look at cases when there are differences between these three methods. Method A will tend to cause propagation errors because P_i is never compared directly to P_0 in the i^{th} frame. It is therefore possible that at each of the intermediate frames k where $(0 < k < i)$, the best guess for P_k deviates further and further from the reference training point P_0 while still remaining close enough to its immediate temporal neighbor P_{k-1}. The reason why we need to avoid propagation error is because we assume our reference point P_0 and P_i belong to the same region or class. Thus with an increase in propagation error, there is an increased risk that our assumption is not valid. In methods B and C the risk of propagation error has been diminished, specifically because we compare

our best match to our reference point P_0. The difference between these two methods is the location of the center of the search range in the i^{th} frame. Method B places the center around (x_{i-1}, y_{i-1}, t_i) and Method C around (x_0, y_0, t_i). If the search range is large enough, at the cost of increasing the amount of computation, the two methods perform similarly.

3.3.2 Region-Based Tracking

An alternative tracking approach to point-based tracking, in which the underlying motion model estimation is block matching, is region-based tracking. In the region tracking method we assume a rigid body motion for an entire region. We combine the optical flow motion information for all of the training points in each region ω_i and apply an affine motion estimate. All of the training points at each region are warped according to their prescribed affine motion estimate in order to predict the location of the training points at future frames. However, in practice we found that user defined regions often do not exhibit rigid body motion and therefore the tracking method used in our segmentation experiments was limited primarily to the point-based method.

3.4 PDF Estimation

Recall in Section 2.2 we provided a general background on two types of probability density function (PDF) estimators. Parametric estimation assumes the shape of the PDF is known, and we solve for parameters that describe the assumed shape. Non-parametric estimation does not assume the shape of the PDF, and we solve for the PDF using numeric methods. Sections 3.4.1 and 3.4.2 discuss how these parametric and non-parametric PDF estimation techniques can be applied to our image segmentation framework.

3.4.1 Parametric

In our segmentation problem, we choose to use a parametric estimation method in which the assumed shape of the PDF for each attribute is a multimodal Gaussian distribution. Equation 2.15 (repeated here) shows the multimodal Gaussian equation.

$$P(x) = \sum_{i=1}^{M} \frac{w_i}{\sqrt{2\pi}\sigma_i} \exp\left(\frac{-(x-m_i)^2}{2\sigma_i^2}\right) \tag{3.30}$$

where x is the random variable we are trying to characterize, M refers to the number of modes, and (w_i, m_i, σ_i) (for $1 \leq i \leq M$) represent the weight, mean and standard deviation of each mode within the multimodal PDF. In stage III of Figure 3-2 our input is the sample training values of each image attribute for each region. If there are a total of F dimensions for all our image attributes and R regions or classes defined by the user, then for the training samples of each attribute (per each region) we solve for the multimodal PDF parameters as described by Equation 3.30 (producing a total of FxR sets of parameters to solve for). Each set of parameters is solved separately, thus we are implicitly assuming an independent set of features. We implement the EM algorithm both with deterministic annealing and cross validation as described in Section 2.2.3.

The PDF estimation is made from the training samples provided by the user in the first frame. The PDF estimate is updated at each successive frame by using the result of the tracking process to determine the location (and thereby the image attribute values) for the training points. Alternatively, we can simply allow for the PDF estimate (of each feature in each region) to remain unchanged for the duration of the sequence. However, some of these attributes (specifically motion and position information) can change significantly over the span of a sequence regardless if its length. A process that tests whether or not the PDF estimate should be updated would be most useful. This would require that the segmentation be solved before the PDF estimation, whereas in our process sequence these steps are indeed reversed. The next best alternative to designing an automatic process that decides when the PDF estimate should be updated is a middle ground between updating all the PDF

estimates at every frame and not updating any of them at all. Namely, those attributes that have a tendency to change more often are updated more frequently. In our experiments, for example, we update motion and position information at each frame, but texture and color PDF estimates are based only on the training points from the initial frame.

Multimodal Gaussian PDF estimation is not the only parametric model that can be used. Initially, a unimodal Gaussian model was implemented, thereby removing the need for any EM algorithm. The only parameters that were calculated were the mean and standard deviation for each feature at each region. However, many of the features in different regions would often exhibit heterogeneous behavior. So, while in theory any "well behaved" PDF can be expressed as a sum of Gaussian PDF's, in practice we found that since the regions were both small enough and similar enough, generally their PDF could be modeled with a sum of one to five Gaussians. Figure 5-1 in Section 5.1 shows the "actual" PDF for a variety of features and image sequences. The actual PDF is based on what is referred to as **ground truth**: a manual segmentation used to evaluate the performance of the machine-driven segmentation. While some features (position, for example) tend not to be multimodal Gaussian, there is no other *a priori* assumption that would work well unless a non-parametric model is used (See Section 3.4.2).

3.4.2 Non Parametric

Most non-parametric PDF estimators typically use a histogram method or some variation thereof. Direct histogram methods classify each data point into one of N bins, where the bins are uniformly spaced along the domain of the possible values of the random variable. There are many potential problems which arise in direct histogram methods, especially for our application in which the data samples we have are sparse relative to the entire population of data points remaining to be labeled.

One problem is that there is a sparse number of data samples. As a result, there is a strong likelihood that either many of bins in the histogram distribution will remain empty, or the bin size will be extremely wide and the histogram distribution will

be sparsely sampled. Another problem is that the dynamic range of the training data samples might not be completely indicative of the dynamic range of the true underlying data set; thus it is difficult to set the bounds on the histogram estimation.

A common non-parametric alternative to direct histogram estimation is one that applies a filter-like kernel centered around each data sample value. **Parzen** estimators (discussed in [95]) estimate a PDF as:

$$\tilde{p}_y(\hat{y}) = \frac{1}{N}\sum_{i=1}^{N}\gamma(\hat{y} - y^{(i)}) \tag{3.31}$$

$$p_y(\hat{y}) = \lim_{N\to\inf} E[\tilde{p}_y(\hat{y})] \tag{3.32}$$

where N is the number of training samples, $\gamma(y)$ is the filter kernel function, $y^{(i)}$ is the set of observed training sample values ($1 \leq i \leq N$), $E[]$ is the expectation or mean value function, $p_y(\hat{y})$ is the true underlying PDF, and $\tilde{p}_y(\hat{y})$ is the estimated PDF. One problem with the Parzen estimation method is that the equality between $\tilde{p}_y(\hat{y})$ and $p_y(\hat{y})$ only holds if N is large, whereas in our case this will not always be true. Another problem lies in deciding the kernel function $\gamma(y)$. The kernel function typically has a Gaussian shape and is sometimes triangular or rectangular. The main difficulty is deciding the characteristic width of the kernel. (This implicitly sets the height of the kernel, because we constrain the kernel function to integrate to 1.) For a Gaussian kernel the variance corresponds to the characteristic width. The narrower the width, the more emphasis we are placing on each training sample value to estimate the true underlying PDF. While a wider kernel results in a "spreading out" of the PDF estimate and potential "smearing" of true peaks. So the resulting PDF estimation is very sensitive to the choice of width of the kernel. In our segmentation experiments we focus only on a parametric PDF estimation and thereby avoid the sensitivity of a non-parametric estimation.

3.5 Classification From PDF Estimation

The final stage in the segmentation process (as per Figure 3-2) is the labeling or classification of all of the remaining unlabeled pixels given a PDF estimation for each feature at each region and given a multi-dimensional feature value $\bar{f}(x,y)$ at each pixel location (x,y). This type of problem is often referred to as **M-ary detection** or **hypothesis testing**, because in the system we need to choose one of M hypotheses. In our case, M corresponds to the number of user-defined regions and each hypothesis H_i where $(1 \leq i \leq M)$ refers to the hypothesis that pixel (x,y) belongs to region R_i. We define the conditional probability that hypothesis H_i is true given the multi-dimensional feature observation as $P(H_i|\bar{f}(x,y))$. This is called an *a posteriori* conditional probability. For shorthand notation, we will rewrite this conditional probability as $P(R_i|\bar{f})$, referring to the probability that a pixel belongs to region i, given the feature vector.

The common solution to hypothesis testing is called the **Maximum a posteriori** (MAP) criterion, or the **likelihood ratio test** (LRT) [95], [31]. At each pixel location, we want to find i corresponding to region R_i such that $P_{R_i|\bar{f}}$ is a maximum (i.e. the conditional probability that the current pixel belongs to region i given feature vector $\bar{f}$ must be maximized.) This estimate is correspondingly called the maximum *a posteriori* or MAP estimate, since i maximizes the posterior probability $P_{R_i|\bar{f}}$. Using Bayes' rule:

$$P(R_i|\bar{f}) = \frac{p_{\bar{F}|R_i}(\bar{f}|R_i) \cdot P(R_i)}{p_{\bar{F}}(\bar{f})} \tag{3.33}$$

where $p_{\bar{F}|R_i}(\bar{f}|R_i)$ represents the multi-dimensional PDF estimate of the image attributes of region i, $P(R_i)$ is the *a priori* probability that a pixel belongs to region i, and $p_{\bar{F}}(\bar{f})$ represents the unconditional joint distribution of the image attribute space, though there is no need to calculate this directly. From Equation 3.33, maximizing $P_{R_i|\bar{f}}$ with respect to i is the same as maximizing:

$$p_{\bar{F}|R_i}(\bar{f}|R_i), \tag{3.34}$$

since we assume the priors $P(R_i)$ are all equal, and since the denominator of Equation 3.33 does not depend on i. If we further assume that the features are uncorrelated since they each represent different aspects of the image (*e.g.* motion, texture, color, and position) the calculation of Equation 3.34 becomes a joint product of the one-dimensional PDF estimates of each feature for each region R_i. Each one dimensional PDF estimate is evaluated at the experimental value of the random variable $f_j(x, y)$. That is, for the j^{th} feature we look at the PDF estimate for that particular feature and evaluate what the PDF value is which corresponds to the value of the feature at the pixel location (x, y). Finally, the pixel sample is assigned to the region which corresponds to the largest product.

Let us further explore Equation 3.34 in the specific case where the PDF estimates for each region is an n-dimensional jointly unimodal Gaussian, with n-dimensional mean $\bar{\mu}_{f_i}$ and nxn covariance matrix Λ_{f_i} (where i is an index variable over the number of regions).

$$p_{\bar{F}|R_i}(\bar{f}|R_i) = \frac{\exp\left(-\frac{1}{2}(\bar{f} - \bar{\mu}_{f_i})^T \Lambda_{f_i}^{-1}(\bar{f} - \bar{\mu}_{f_i})\right)}{(2\pi)^{n/2}|\Lambda_{f_i}|^{1/2}} \tag{3.35}$$

We can maximize Equation 3.35 with respect to i by taking the log, since the log function is a monotonically increasing function. The result is:

$$-\frac{1}{2}\left[n(\log(2\pi)) + \log(|\Lambda_{f_i}|) + (\bar{f} - \bar{\mu}_{f_i})^T \Lambda_{f_i}^{-1}(\bar{f} - \bar{\mu}_{f_i})\right] \tag{3.36}$$

The first term of Equation 3.36 ($n\log(2\pi)$) is the same for all hypotheses. If we include the additional assumption that the features are independent, Equation 3.36 can be further reduced to

$$-\frac{1}{2}\left[\log(\prod_{j=1}^{n}\sigma_{j_i}^2) + \underbrace{\sum_{j=1}^{n}\frac{(f_j - \mu f_{j_i})^2}{\sigma_{j_i}^2}}_{\text{Mahalanobis Distance}}\right] \tag{3.37}$$

where f_j is the value of feature j at a particular pixel location, μf_{j_i} and $\sigma_{j_i}^2$ are the mean and variance of feature j in region i, and n is the number of dimensions

of the feature space. Note the second term in Equation 3.37 is known as the **Mahalanobis distance**. So when the minimum Mahalanobis distance is used as the decision criterion, the $2\sum_{j=1}^{n}\log(\sigma_{j_i}) = \log(\prod_{j=1}^{n}\sigma_{j_i}^2)$ term is effectively ignored.

Associated with the classification of each sample is a certainty measurement for that classification. The certainty measurement used is a function of the ratio between the likelihood of the second-most likely region and the most likely region. Accordingly, if the likelihood of the most likely region is very close to the likelihood of the next most likely region, then there is a low certainty, since it is difficult to discern between the two multi-dimensional distributions. Similarly, if the likelihoods of the most likely and second-most likely regions are widespread, the certainty is high. The following equation is used to calculate a numeric value for the certainty:

$$c(x,y) = 1 - \frac{L_2}{L_1} \tag{3.38}$$

where c is the certainty evaluated at each pixel in the image sequence. L_2 is the likelihood (or log likelihood) which corresponds to the joint product of the one-dimensional PDF evaluated at the feature value for the region corresponding to the second most likely region. L_1 is the likelihood (or log likelihood) which corresponds to the joint product of the one-dimensional PDF evaluated at the feature value for the region corresponding to the most likely region. Note that $L_2/L_1 <= 1$ since $L_2 <= L_1$, which means the certainty c is always between 0 and 1.

Chapter 4

Segmentation Implementation

This chapter describes in detail the C program specifications that have been compiled and tested on DEC alpha (and a DEC 5000) series processors, in an X/Unix environment and system configuration. Image data files are stored in a "datfile" format which is essentially a directory containing a **data** file, a **descriptor** file, and in some cases, a **lut** file. The data file contains only the raw image data. The descriptor file contains header information including the number of rows, columns, channels, frames (also referred to as data sets), data type (*e.g.* integer, float, double, etc.), and often some history or comment which describes what the data file is and/or how it was created. The lut file is optional and contains look up table information in the event one would want to map index values (such as region numbers) to varying colors or gray scale values. In this chapter I refer to a **mask file** as a type of data file that when displayed, shows each region as a separate color. In actuality, the mask file contains a lut file, and has a pixel value in the range of 0 and N, where N is the number of regions in the image. The value of 0 is displayed as black, and corresponds to an unclassified pixel. The values of regions 1 through 10 are displayed with the following RGB values. (Note: these LUT values are for video monitor medium. For print medium, black and white are reversed.)

Region #	Color Name	Red value	Green value	Blue value
0 (unlabeled)	black	0	0	0
1	white	255	255	255
2	red	255	0	0
3	green	0	255	0
4	blue	0	0	255
5	yellow	255	255	0
6	magenta	255	0	255
7	cyan	0	255	255
8	grey	128	128	128
9	salmon	255	128	128
10	light green	128	255	128

Table 4.1: LUT values for mask files, up to 10 regions.

4.1 Block Diagram

Figure 4-1 shows the entire block diagram of the segmentation process. (Note this is identical as Figure 3-2, but simply re-shown here.) Recall that the overall process can be broken down into four stages. The first, labeled "Feature Calculation" deals with acquiring and calculating the feature space from the image sequence. The second stage, labeled "Training and Tracking" deals with acquiring the training data and tracking the position of the training data over time. The third stage, labeled "PDF Estimation" deals with estimating the probability density function of every feature in each region. Finally, the fourth and final stage, labeled "Classifier" deals with the decision rule used to classify all the data points into the most likely class, given the multidimensional feature calculations and PDF estimations.

Thus the "input" for the entire system is simply an RGB image sequence (where the R, G, and B values are defined at each pixel sample location) as well as a set of user-defined training points. These training points label representative samples from each of the regions which are defined by the user. Thus, the number of regions as well as the number of training samples are not predetermined amounts Typically the system has been run with three to ten regions, with 100 to 3000 training samples per region on an image size of about 300 rows by 300 columns by 20 frames. Given only

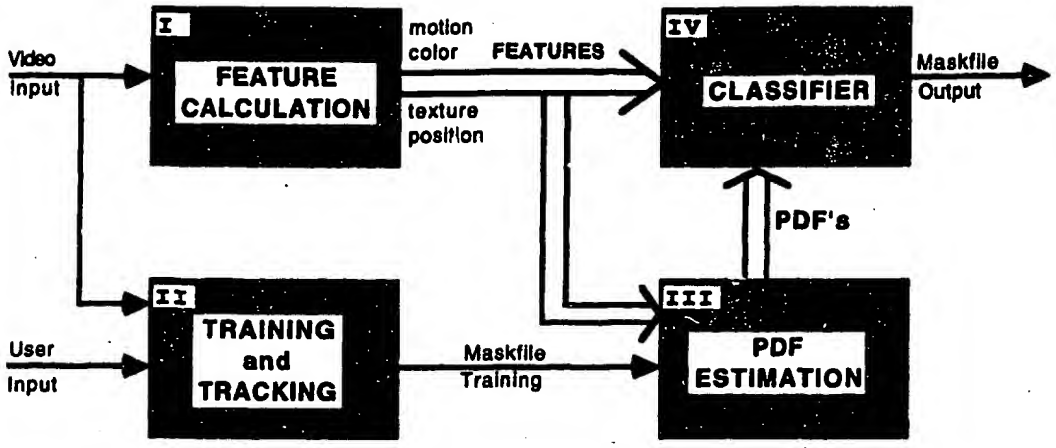


Figure 4-1: General block diagram of segmentation algorithm.

about 0.5% of the image as training, we'd like to classify up to 100% of the image sequence as accurately as possible.

4.2 Feature Calculation

The first stage of the block diagram above deals with extracting the "feature" or image attribute information (such as motion, texture, color and position) given the information contained in the input RGB image sequence. The feature transformation step can sometimes be computed in a simple direct method as in the case of color and position attributes, or it can require a more complicated computation method, as in the case of texture and motion attributes.

4.2.1 Texture

For the experiments described in Chapter 5, we used three different methods to calculate texture characteristics. These three methods are local statistics, multiresolution statistics and simultaneous auto regressive (SAR) parameters.

In the local statistics method we compute the mean and standard deviation of the luminance channel only (or optionally of each channel in a color format). The number of dimensions this feature produces is two if the input is luminance only, or six if the input is color. One parameter of the local statistics program is the window size over

which to compute these local statistics. Typical values range between 3 pixels to 15 pixels.

In the multiresolution statistics method we compute the mean and standard deviation typically of the luminance channel (optionally of each channel in a color format) and typically at three local window sizes (such as 3,9, and 15 pixels). This method produces two dimensions of texture per window size when the input is the luminance channel, and it produces six dimensions of texture per window size when the input is the luminance channel. Since we typically calculate the texture statistics at 3 levels we do not compute multiresolution statistics for color because that would produce too many (18) dimensions of the texture feature. In fact, because the texture values at different levels are correlated we only use the standard deviation component of the texture corresponding at one window size (typically the middle window size) along with the luminance component at all three window sizes. Thus in our experiments, the multiresolution texture feature is a four dimensional feature.

The simultaneous autoregressive SAR method produces a five dimensional feature. Four dimensions correspond to the weighting coefficients of the immediate neighbor values. The other dimension corresponds to the error in representing a pixel value by the weighted sum of its neighbors' values. Recall that the best estimate of these parameters comes from finding a LSE over a N by N region. In the SAR program, this parameter N can be specified upon executing the program. Typical values of N were between 3 and 13.

4.2.2 Color

The color formats that have been experimented include RGB, YIQ, and LAB. Each of the color features we estimate produce a three dimensional feature vector and are calculated using a straightforward conversion program. To produce RGB data no transformation program is required since the original input image sequence is typically in RGB format. To calculate the YIQ feature vector a simple linear transformation program is applied to convert RGB to YIQ. To calculate the LAB feature vector a transformation program is applied. The program applies an intermediate

CIE transformation that applies the appropriate reference chromaticity coefficients stored internally.

4.2.3 Motion

Of the motion estimation techniques described in Section 3.2.1, the optical flow Lucas and Kanade method, and the optical flow Horn and Schunck methods were implemented. In both cases, the programs calculate only one motion field, consisting of a horizontal and vertical component, for a pair of input frames. These input frames typically correspond to two sequential frames, luminance channel only.

The Lucas and Kanade method produces two dimensions corresponding to the horizontal and vertical motion. The program takes as input the luminance channel of two consecutive frames of an image sequence. Two optional parameters can be supplied to the algorithm: the first parameter controls if the spatial derivative should be calculated using 2 or 4 local pixel values, and the second parameter specifies if a Gaussian spatial prefilter (the width of the Gaussian filter can also be specified) should be applied to the image when calculating the optical flow.

The Horn and Schunck method also produces two dimensions corresponding to the horizontal and vertical motion. The program has the same parameter format as the Lucas and Kanade program plus it contains two additional optional parameters. One optional parameter of the Horn and Schunck (not found in Lucas and Kanade) method controls how much weighting (the α coefficient of Equation 3.13) is applied to the global smoothness constraint The other optional parameter specifies the maximum number of iterations the program should go through before halting.

4.3 Training and Tracking

Stage II of the block diagram in Figure 4-1 deals with acquiring "supervised" or "training" samples for each of the regions in the image sequence. This stage has a training component and a tracking component. The first deals with acquiring the training points from the user, and the second deals with tracking the training points

or estimating the position of those training points in future frames. The tracked position at each frame is used as the training points to estimate the PDF of each of feature in each region. In most of the segmentation experiments, the tracking method used was a block matching, because using the region matching method seemed to have caused propagation errors, due to inaccurate affine parameters.

4.3.1 Training

The training program is quite simple to understand conceptually. It uses an X-interface to allow the user to highlight representative points from each region. By dragging the mouse, the user controls which points accumulate into the training set of the current region. And by clicking a mouse button the user indicates the points should be accumulated into the next region. The optional parameters control the format of the potential output files, and indicate some display characteristics so the algorithm could work on different types of displays. The two formats of the output are useful depending on whether or not the location of the training data will be displayed. A **mask datfile** format is used if the the training data will be displayed. In this case, the format of storing the training data is to use the value of 1 to R, corresponding to the region number, (where R is the total number of regions defined by the user) at every sample point that is a training data point, and a value of 0 where there is not a training data point. A look up table as described in Table 4.1 is helpful for viewing the training data by transforming each of the values 0 to R to a distinct color. An **index datfile** format is used if the the training data will not displayed. In this case, the format of storing the location of the training data is in a columnar format. Since a majority of the image sample points are not training points. we only need to record the column, row, and region information of each training data point in this index format. As a side note, I've also written utility programs to convert between the mask datfile format and the index datfile format. See the end of Section B.3 for a more detailed description of these utilities.

4.3.2 Tracking

Once the training points from the first frame of an image sequence have been acquired, the next step is to estimate the location of these training points at future frames. Conceptually there are two different approaches, given our training points. The first approach, a region-based method, relies on the fact that the training points are already classified by the user, and assumes that all the points in a region have similar motion. Thus the motion estimation information is used from all of the training samples which belong to the same region (for each region), and an affine estimation algorithm is then applied to the motion information, resulting in a 6-parameter affine motion model for each region. The second approach, a block matching algorithm, estimates the motion for each sample individually. Therefore it does not apply a rigid-body constraint implicit in the affine model.

In our experiments, it has been found that the region-based tracking approach (that uses an affine motion model) described in Section 3.3.2 did not perform as well as the point-based tracking approach, (that uses a block-matching motion estimation) so I will describe the point-based tracking program only. The input and output parameter specifications of the program are straightforward. The program takes as input an image sequence file and an index datfile containing the pixel location and region number of the training data. The program produces as output a mask datfile containing the estimated location of the training data at each frame in the sequence. Internal parameters of the algorithm include the block size and search range used in correlating a block in one frame to a block (whose spatial offset is within the bounds of the search range) in the next frame.

4.4 PDF Estimation and Segmentation

The third and fourth stages from Figure 4-1 which deal with the PDF estimation and the labeling (or segmentation) procedures, have been combined into one program. This program takes as input a index datfile and at least one feature datfile or row or column position, and produces as output a mask datfile. The index datfile is provided

at each frame and contains the pixel location and region label of the training data (in the first frame) and the tracking data. (in successive frames) The feature datfile contains the image attribute values, and the program internally estimates the PDF of each feature at each region. The program classifies all of the unlabeled samples to the region corresponding with the maximum likelihood given the PDF estimation, and the feature data.

Some optional parameters of this program include: whether or not to save the certainty measurements associated with the classification at each sample point; a parameter that controls if all points should be labeled or only those that have a certainty measurement above a specified certainty threshold; whether or not to save PDF estimates (calculated as an intermediate step) of each feature in each region; and finally, an optional parameter is used to indicate whether the training and tracking samples should be assumed that they are correctly labeled, (usually the case with the training data, but not necessarily the case with the tracking data) or if they should be relabeled based on the PDF estimation and segmentation process.

There are actually two versions of a program that combines the PDF estimation and segmentation processes. Both versions have the identical parameter specification, but they differ in the way the PDF estimation is implemented. One version implements the basic EM algorithm as described in Section 2.2.2, and the other version implements the EM algorithm with deterministic annealing as described in Section 2.2.3.

4.4.1 PDF Estimation Only

Sometimes it makes sense to breakdown the probability estimation and segmentation process into two pieces. Since the tracking algorithm may have errors in the prediction of locating the training points, then building a PDF estimate from those erroneous training points will disturb the PDF estimation and ultimately cause errors in the segmentation. Better results might be obtained if the PDF estimation is calculated based exclusively on the training points of the first frame. The segmentation can be calculated at any future frame, given the PDF of the first frame. A combination

of these two methods is to update the PDF estimate of only some of the attributes (such as position and motion) and to assume that the other attributes (texture and color) are quasi-stationary. When calculating the segmentation one can therefore pick and choose which PDF estimates are to be used at each frame of the sequence if a PDF estimation program calculates this information separately form the segmentation program.

A program was designed specifically for this purpose. It takes as input a index datfile, and a minimum of one image feature. It produces as output a datfile (or text file) that contains the PDF estimate of each feature in each region. The PDF estimate is stored in a columnar format. One colum corresponding to the domain (or feature) values and one column corresponding to the range (or likelihood) values. There are two versions of the PDF estimation program for the same reasons (described in Section 4.4) there are two versions of the program that computes the PDF estimation and the segmentation.

4.4.2 Segmentation from PDF Files

There is a program that produces a segmentation mask datfile given a PDF datfile or set of PDF datfiles, and corresponding feature files. In this segmentation program, an index datfile is completely optional because the classification at each picel depends only on the value of the multidimensional feature and the multidimensional PDF estimate of each region. The index datfile is optional also because it may be that the PDF estimate was calculated based on training or tarcking points at one frame, and these PDF estimates will be used to calculate the segmentation at a different frame. Other optional parameters to this segmentation program are similar to the optional parameters to the program described in Section 4.4. These parameters include whether to save the certainty measurments, what (if any) is the value of the certainty threshold, and whether to save the PDF estimates.

4.5 Post-Processing

There are a pair of post-processing programs (usually executed together) that can apply an alternate classification rule. The first program unassigns points that have been classified with a certainty measurement below a specific certainty threshold. The certainty measurement is a coefficient between 0 and 1 that estimates how certain we are of our classification decision. By systematically removing samples with low certainty values, we have designed experiments that measure the tradeoff between percentage of points labeled vs. the percentage of accurately labeled points. We expect to notice that as more of the uncertain points are unlabeled, a greater percentage of the remaining labeled points are correctly labeled.

The second program usually takes as input the mask datfile that was produced by the declassification program above. More generally, the program may take as input a mask datfile with almost all of the points labeled (but some of them unlabeled) and assigns the remaining unlabeled points to the region corresponding with the most popular region amongst its neighbors. The method of classifying these unlabeled points is known as a "K-nearest neighbor" method (using the pooled method of KNN) and is described in a bit more detail in Section 2.3.2. The parameters to the program are quite straightforward, an input mask datfile is required, and an output mask datfile is produced. The output datfile contains a labels at every point, where as the input may have points that are not labeled. An optional paramater to the program specifies what the neighborhood size the KNN algorithm should search in order to label the unlabeled points.

Chapter 5

Segmentation Experiments and Results

Now that we have discussed the "backbone" of the segmentation algorithm, and have explained the parameter variations on the different "modules", let us take a look at a variety of experiments and results. In this chapter, we will show example image sequences, as well as the user-supplied training points, the location of tracked training points, the PDF estimation of the multi-dimensional feature space, and the final segmentation.

We will focus on three short sequences, "Gymnast" (GYM), "Good Morning Vietnam" (GMV), and "A Fish Called Wanda" (FISH). For these three sequences, we have computed a ground truth based upon a manual pixel by pixel labeling for 10 frames. We therefore can compute the accuracy of the many parameter variations relative to the ground truth. We also give some examples of our segmentation scheme for sequences in which no ground truth has been calculated, and therefore we must use a subjective measurement to evaluate the performance of the segmentation.

5.1 PDF Estimation

Before we demonstrate the segmentation results, let us examine the performance of the probability density function estimation module. Because the multi-dimensional

features are treated separately, there are typically about eleven PDF estimations for each user-defined region, at each frame. And since there are many possible ways of calculating each of the features, it would be impossible to show all these PDF estimations.

Let us look at the actual statistics of one entire frame of an image sequence, for each feature. In Figure 5-1 we look at the "actual" PDF of one frame of the GYM sequence (frames of the original image sequence are shown in Figure 5-6). The image size is 232 rows by 348 columns, and thus contains 80736 samples. Before examining the statistics of the attributes in the image on a region by region basis, let us first examine the statistics of each of the color, motion, texture, and position attributes over the entire first frame. The features presented are: color, LAB format; texture, multiresolution statistics. Namely LM1, LM2, LM3 correspond to the local mean at resolution size 3x3, 9x9, and 15x15, respectively, and LSD1 corresponds to the local standard deviation at 9x9 resolution; motion, optical flow using the Lucas and Kanade method; and finally position is simply row and column location. With the exception of the row and column position features, the remaining features fit well to a multimodal Gaussian PDF, with few (up to 5) modes. In theory, any PDF can be described by a weighted sum of Gaussian PDF's, given enough of them. However, in our experiments we want to keep that number of Gaussian modes low, though we don't want to explicitly restrict to be one Gaussian.

In Figures 5-2 through 5-4, we examine the performance of the multimodal PDF estimation using the EM algorithm as described in Section 2.2.3. We examine the PDF estimation of all the features, for one region in one frame of the GYM sequence. Specifically, we look at Region 7, which corresponds to the main gymnast in the foreground of the image. The three figures compare pairwise, the true PDF, the estimated PDF, and the training data. The "true" PDF is based on the statistics of Region 7 from the manual segmentation (shown in Figure 5-6). The "estimated" PDF is based on the estimation using the EM algorithm. and the "training" data is a histogram based on the user supplied training sample points. (The training data mask file is shown in Figure 5-7).

Figure 5-2 compares the true PDF vs. the estimated PDF. Note that the estimated PDF is typically smoother, since we are limiting it to a small number of modes. Figure 5-3 compares the true PDF to a histogram of the training data. There were 415 training data sample points, so instead of showing the exact value of all 415 points, a histogram (of 10 bins) is used to demonstrate the relative frequency of occurrence of the different values. Finally, Figure 5-4 compares the estimated PDF to the histogram of the training data.

Figure 5-5 compares the estimated PDF vs the true PDF. But instead of looking at all 11 features for one particular region, as in Figure 5-2, here we look at one particular feature, LM1, or local mean 3x3 resolution, over all 7 regions of the first frame of the GYM sequence. In this example, most of the estimated PDF's compare rather closely to the true PDF except for Regions 1 and 2, corresponding to the light in the top left corner and the balance beam. The discrepancy in the estimate of Region 1 is mostly due to the fact that the entire region corresponding to the light contains very few samples, and the training for that region contains even fewer samples. A second discrepancy stems from the fact that the particular feature being estimated is the local mean, and the training samples for region 1 were chosen predominantly from the center of the region. Now the "light" region (Region 1) has very high luminance values, and the pixels on the border outside the light are very dark. The values of the local mean feature along the border of the light region are therefore predominantly lower than the values of the local mean feature in the center of the light region. Note that in Figure 5-2, the true PDF plot corresponding to Region 1 has a large contribution of values in the lower range, (50-100) but the estimated PDF does not catch these values because few or no training samples were chosen with values of the local mean in that lower range. The discrepancy in Region 2 (the balance beam) is also due mostly to the fact that the user training data was not exactly representative of the true region, though that is a little less evident by looking at the location of the training data samples shown in Figure 5-7.

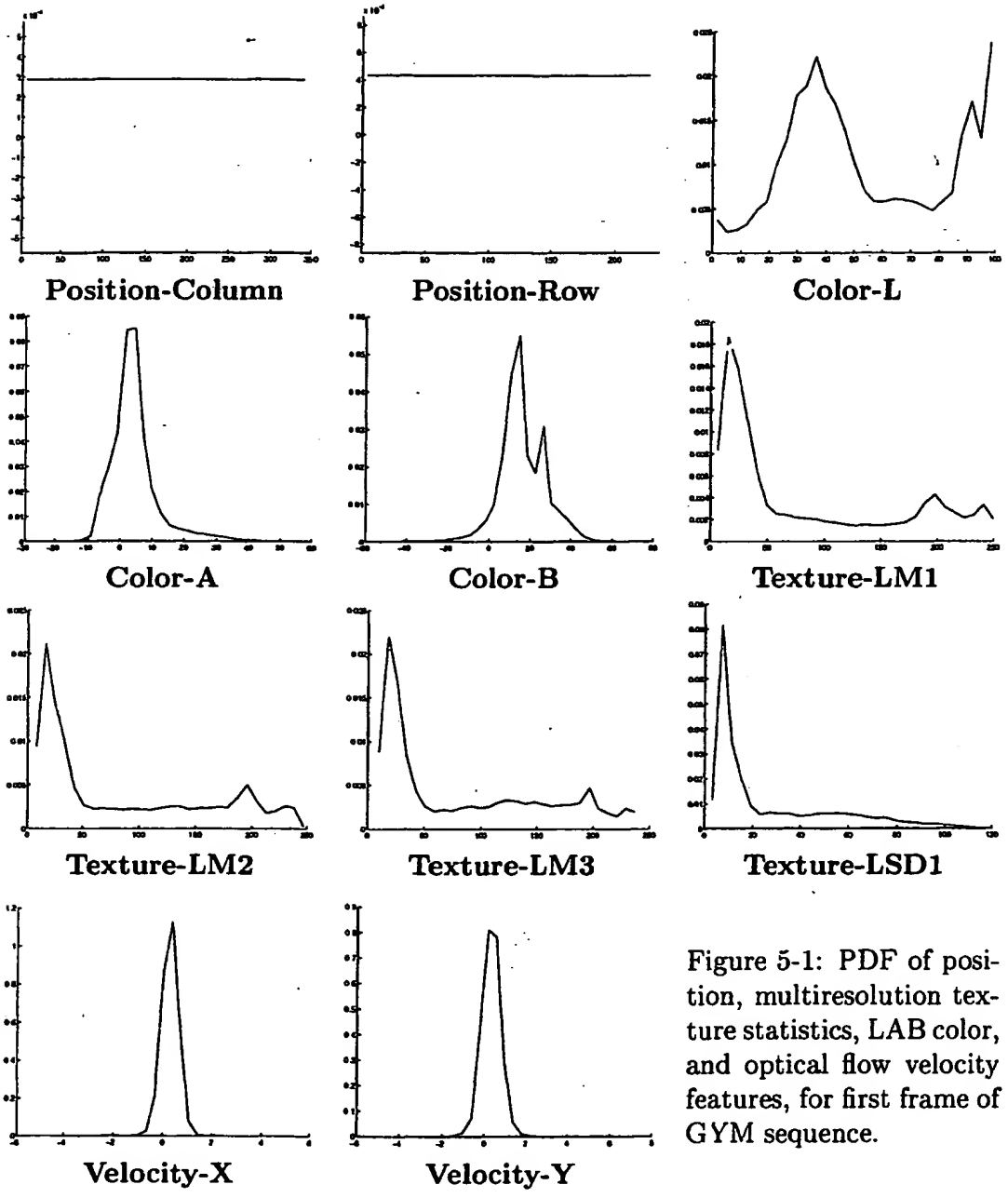


Figure 5-1: PDF of position, multiresolution texture statistics, LAB color, and optical flow velocity features, for first frame of GYM sequence.

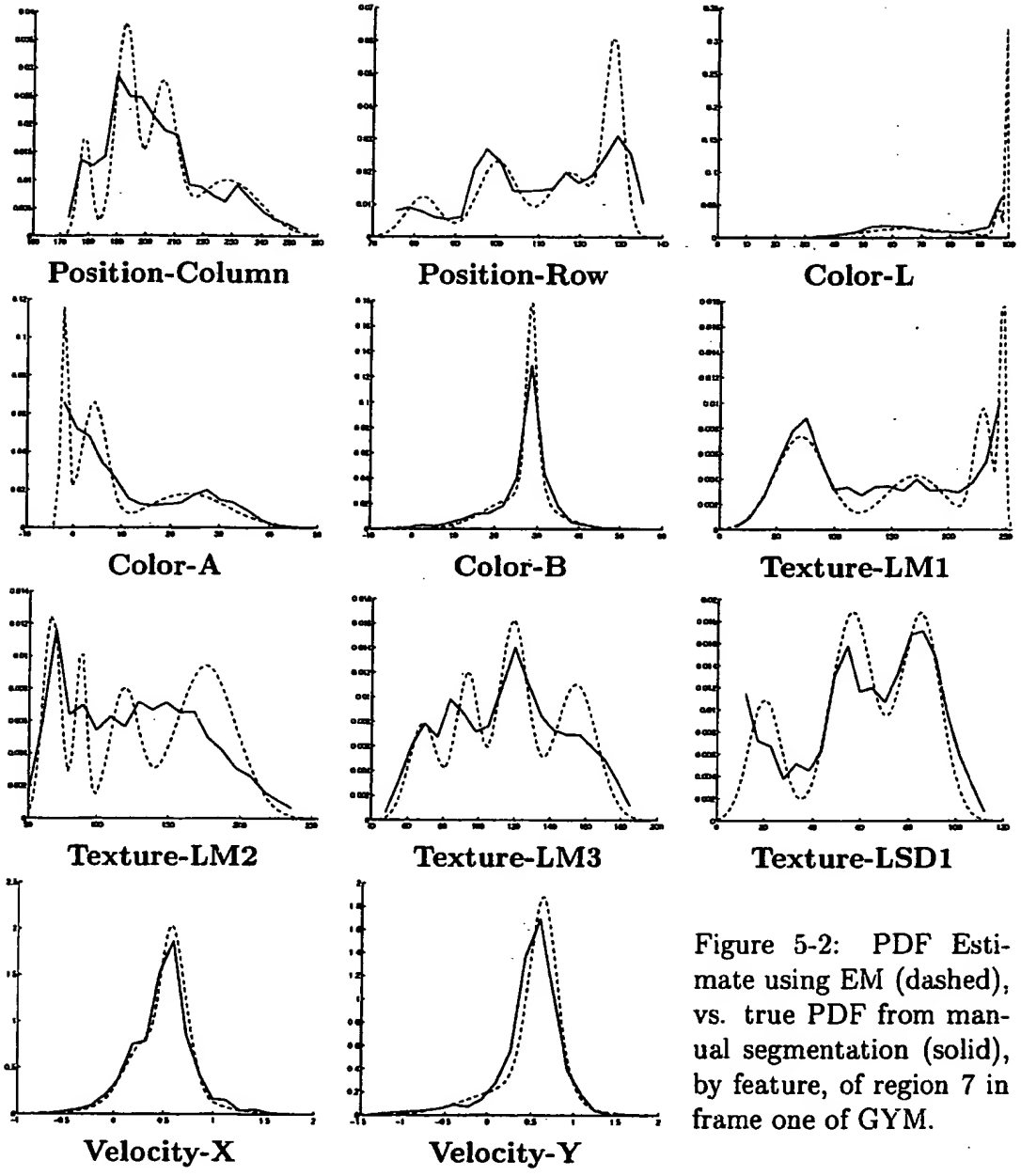


Figure 5-2: PDF Estimate using EM (dashed), vs. true PDF from manual segmentation (solid), by feature, of region 7 in frame one of GYM.

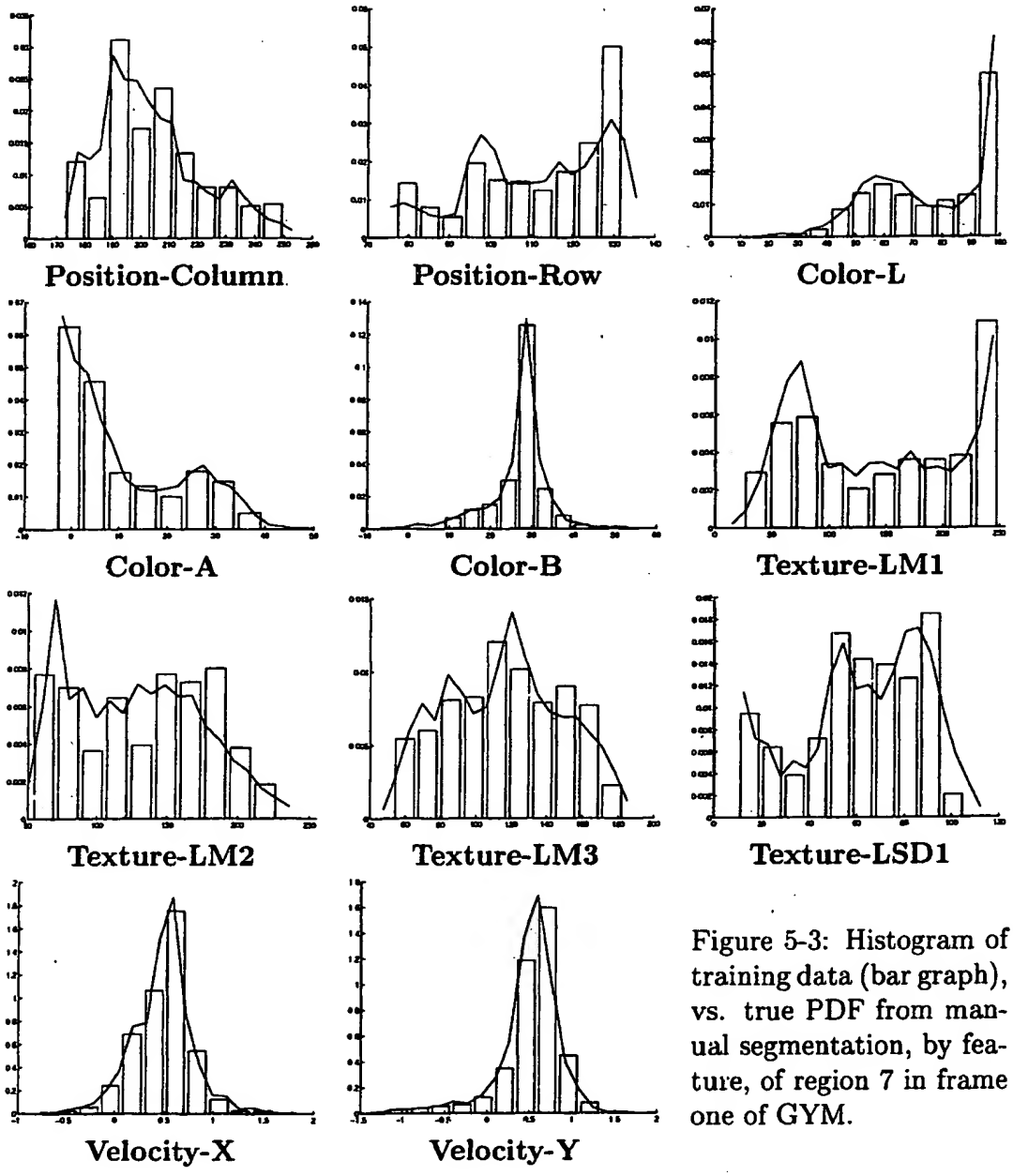


Figure 5-3: Histogram of training data (bar graph), vs. true PDF from manual segmentation, by feature, of region 7 in frame one of GYM.

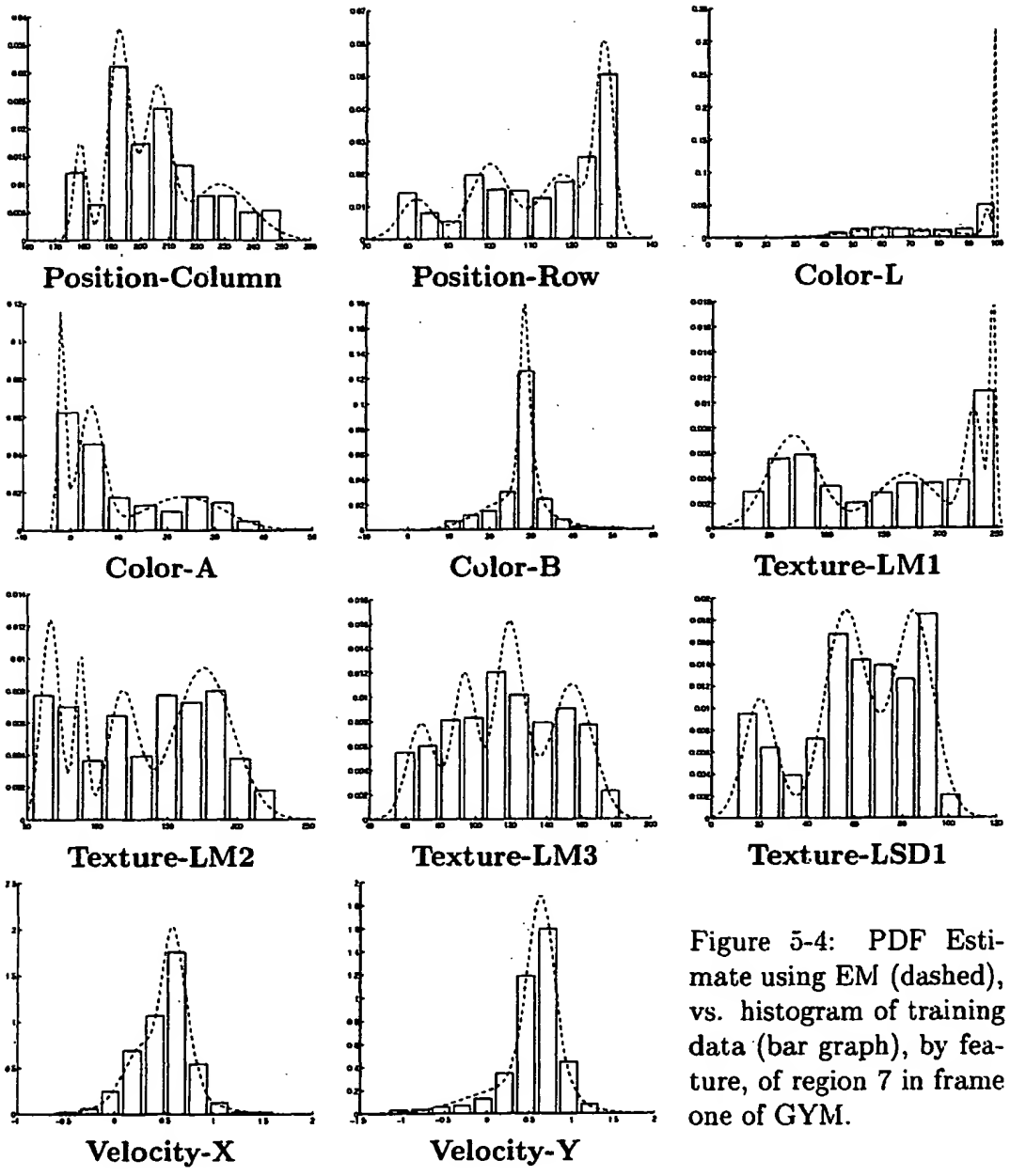


Figure 5-4: PDF Estimate using EM (dashed), vs. histogram of training data (bar graph), by feature, of region 7 in frame one of GYM.

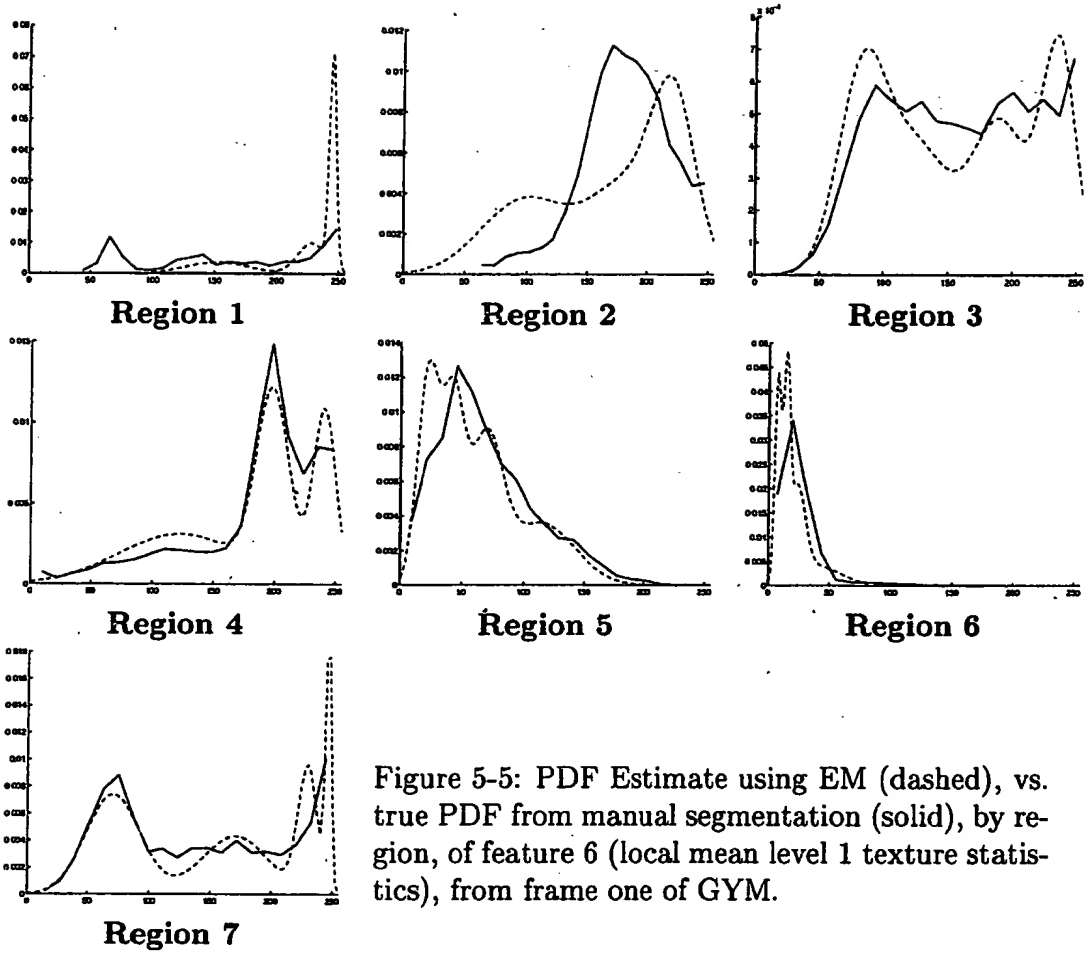


Figure 5-5: PDF Estimate using EM (dashed), vs. true PDF from manual segmentation (solid), by region, of feature 6 (local mean level 1 texture statistics), from frame one of GYM.

5.2 Test Images

For each of the 3 sequences, GYM, GMV, and FISH, selected frames of the original sequence as well as the manually segmented mask file are shown. I will also show several tables that show the "objective" performance, relative to ground truth (manual segmentation), of about 18 different combinations of parameter options for the segmentation process. Tables 5.2 and 5.3 show an index of these parameter choices, and a list of experiments for which the performance is evaluated.

5.2.1 GYM Sequence

The Gymnast sequence is a relatively fast moving sequence, that zooms into the gymnast (in the foreground) in about 20 frames. In our experiment we only focus on the first 10 frames of that sequence, which still contains quite a bit of motion from the gymnast herself.

Figure 5-6 shows frames 1,4,7, and 10 of the original sequence, as well as frames 1 and 10 of the mask file of the manual segmentation. Recall that Table 4.1 shows the values of up to 10 regions (with the minor exception that black and white, regions 0 and 1 are reversed in print medium vs monitor display medium). The user selected a total of 7 regions. Region 1 (black) corresponds to the light in the top right corner, visible only frames 1-4. Region 2 (red) corresponds to the balance beam. Region 3 (green) corresponds to gymnasts in the background. Region 4 (blue) corresponds to the mats. Region 5 (yellow) corresponds to the lower audience seats. Region 6 (magenta) corresponds to the upper audience seats and most of the gymnasium background/ceiling. And Region 7 (cyan) corresponds to the gymnast in the foreground.

Figure 5-7 shows frames 1,4,7, and 10 of the training and tracked data points. The first frame shows the location of the points selected by the user as representative points from each user-defined region. All the remaining future frames (shown in the figure are only frames 4,7, and 10) use the tracking process, as described in Sections 3.3 and 3.3.1 to estimate the location of "training" data at each successive frame.

Table 5.1 shows the number of training points per frame, and also what percentage

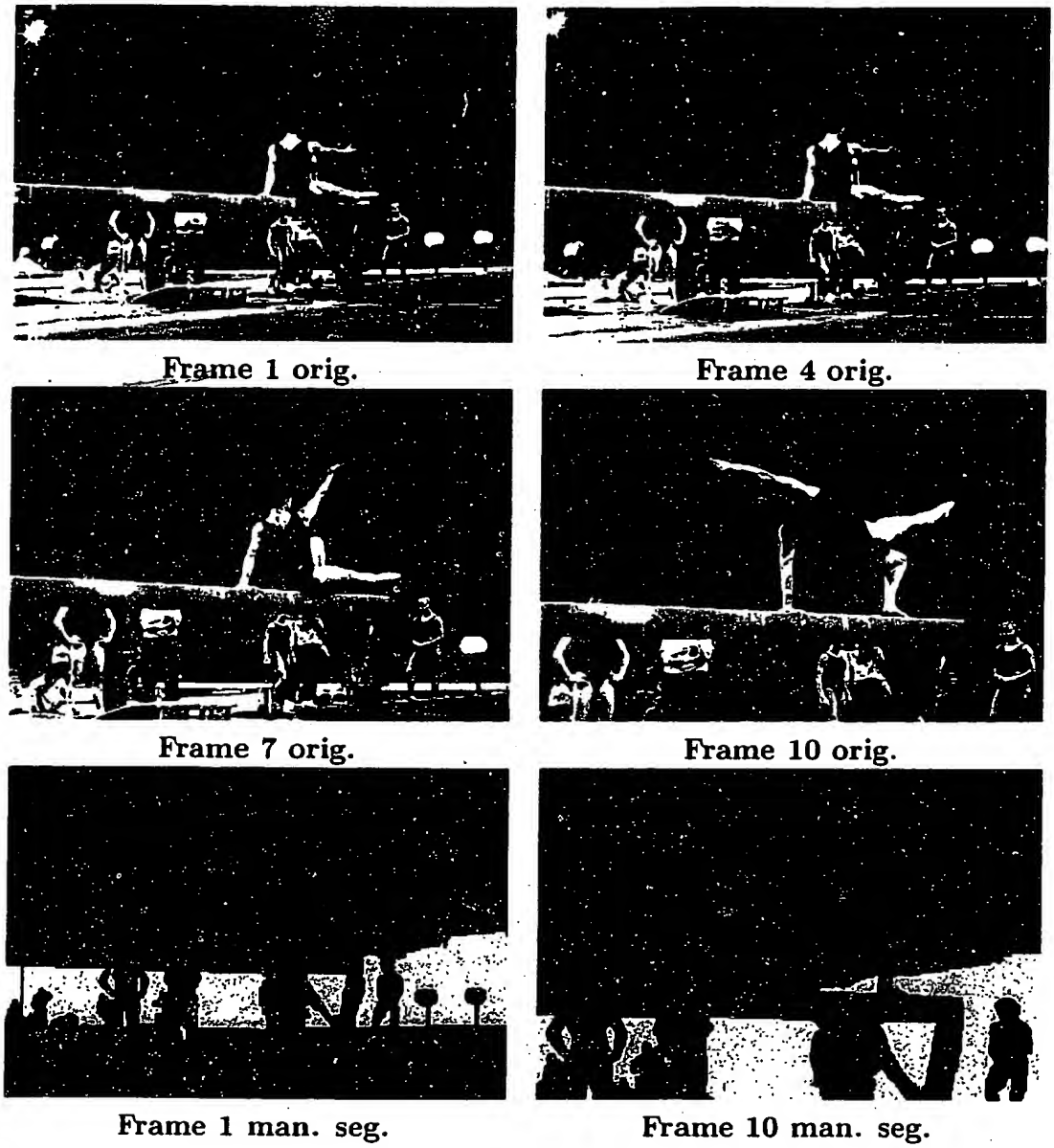


Figure 5-6: Original and manually segmented frames from GYM sequence.

of those training points are accurate with respect to the ground truth data. Note that the user provides training points only for the first frame. So that over a 10 frame sequence, the segmentation scheme attempts to label 807360 samples (10 frames by 232 rows/frame by 348 columns/frame). In the gymnast example, the user provides 4743 samples as training points or roughly 0.587% of the total number of points.

A second point to note is that, even though the accuracy of the training and tracking information is not 100%, we are probably getting a decent PDF estimate of the features for each region. Since the PDF estimation depends more on the representative values for that image attribute, not on the number of "correct" training points vs. "incorrect" training points. So for the first frame, the very small amount of error occurs at portions of the image where the attributes have similar values anyway. So switching those region boundary points in the training data will probably not cause in great impact in the PDF estimation. At future frames, the PDF estimation training data is based on the tracking algorithm, which uses a statistical correlation to determine the new location of samples. Thus even in future frames, errors in training data typically swap values that are statistically similar enough not to create a huge impact in the PDF estimation. Ideally we'd like to minimize or even eliminate error in the training altogether. But in practice there is no guarantee that a user will not provide corrupted training data, probably inadvertently. So for the purposes of the experiment and we performed the the PDF estimation and segmentation using slightly corrupted data.

Let us now discuss the some of the variations of experiments. Table 5.2 shows an index of the many different options for the segmentation experiment parameters, such as how to compute the attributes and which ones to use, as well as how to compute PDF estimation and tracking procedure. Since it would be too numerous to try out every combination of these options, we have selected to discuss the results "objectively" for about 18 experiments, listed in Table 5.3. Objectively means we look at the accuracy of the segmentation relative to ground truth. From those 18 experiments we select only one or two to show sample frames of the mask segmentation output. It would require too much space to show all these results, and since many of

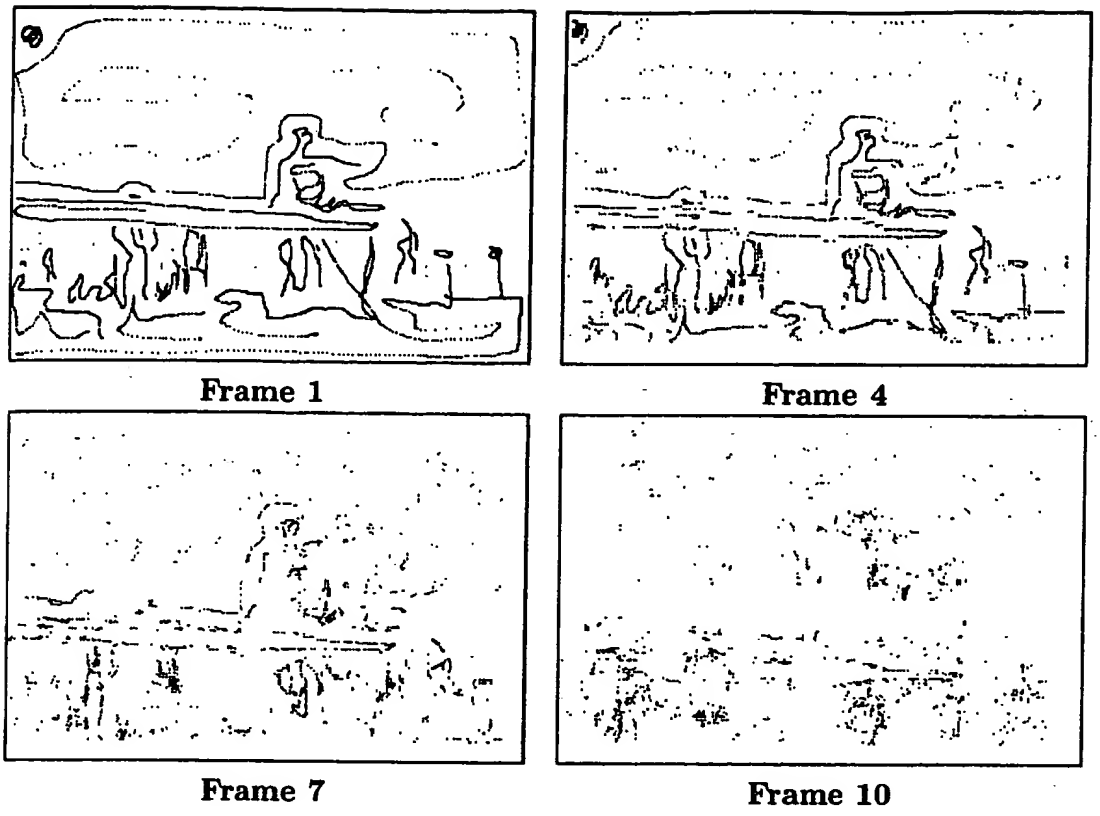


Figure 5-7: Training Data points in Frame 1 of GYM, and tracked location of those points at Frames 4,7 and 10.

Frame	# Train Pts.	% Correct
1	4743	93.612
2	3594	95.604
3	3638	95.547
4	3533	95.754
5	3119	91.632
6	2749	89.851
7	2416	87.666
8	2191	83.341
9	2035	78.231
10	1898	74.131
ALL	**29916**	**90.273**

Table 5.1: Accuracy of training and tracking data points per frame, each frame size is 80736 samples.

Color Index	
key	*represents*
A	RGB
B	YIQ
C	LAB
D	None

Texture Index	
key	*represents*
A	Local Stats
B	Multi Level Stats
C	SAR
D	None

Motion Index	
key	*represents*
A	Horn & Schunck
B	Lucas & Kanade
C	None

Position Index	
key	*represents*
A	Row and Column Location
B	None

Tracking Index	
key	*represents*
A	Block-Matching
B	Region Growing
C	Tracking from Segmentation

PDF Estimation Index	
key	*represents*
A	EM Basic Method
B	EM Deterministic Annealing

Table 5.2: Parameter index key of feature (Color, Texture, Motion, Position) calculation method, tracking method, and PDF estimation method.

them are very similar qualitatively, we select only a couple.

Note that in addition to all the segmentation parameters listed in Table 5.2, there are also sub-parameters for many of the individual programs. For example, if the texture attribute is computed using local statistics, the sub-parameter would be deciding the window size over which to compute the local statistics of the image sequence. There are similar sub-parameters in computing the SAR texture feature, as well as both the optical flow motion features, and the point based tracking algorithm. The sub-parameter that seemed to have the greatest (though not all that large, anyway) impact is the window size on the local statistics texture feature. Typical values of the block size are 5 or 7 pixels. The corresponding window size is indicated in the list of experiments in Table 5.3.

Table 5.4 shows the performance, measured in terms of percentage of points labeled correctly, with respect to ground truth (manual segmentation), for all 18 experiments. The first column corresponds to one of the experiments described in Table 5.3. The second column shows the percentage of points that are labeled. This value varies

Method	Color	Texture	Motion	Position	Tracking	PDF Est.
I	C	A,5	A	A	A	A
II	C	B	A	A	A	A
III	C	A,5	B	A	A	A
IV	C	B	B	A	A	A
V	A	A,5	A	A	A	A
VI	B	A,5	A	A	A	A
VII	B	A,5	B	A	A	A
VIII	B	A,7	B	A	A	A
IX	B	B	B	A	A	A
X	B	C	B	A	A	A
XI	C	A,5	A	A	C	A
XII	C	A,5	A	A	B	A
XIII	C	B	A	A	A	B
XIV	C	B	B	A	A	B
XV	B	D	C	B	A	A
XVI	B	D	C	A	A	A
XVII	D	A,5	C	A	A	A
XVIII	D	D	B	A	A	A

Table 5.3: List of experiments and the corresponding selection of parameter settings. See parameter index key in Table 5.2.

depending on what the certainty threshold value is. You will see later in Table 5.8 that as the certainty threshold parameter is increased, a smaller percentage of points in the image sequence are classified, but of the samples that are labeled, a higher percentage of the points are labeled correctly. 100% labeled corresponds to a certainty threshold of 0.0. The third column shows what percentage of points were labeled correctly. This percentage reflects a conglomerate total percentage over all 7 regions and all 10 frames. Thus a percentage of 85%, for example means that 686256 pixels out of 807360 were labeled correctly. Recall from Table 5.1 that we start with 4743 user-labeled pixels in our training data.

We can look at the segmentation performance on a region by region basis, and also on a frame by frame basis (or both). Again, though, to do this for each one of the 18 experiments would take too much effort, and space. Instead we will focus on the performance of two of these experiments, Method XIV and Method XVI. Before we do though, it would be worthwhile to note the performance of Method XI and XV, the only two with a percentage less than 80 percent. In Method XI, we choose a different tracking method, where at each frame we select points to track into the next frame based on the segmentation of the current frame. Though this method seems sound in concept, in practice it result in a propagation of error. Specifically, points that were labeled as belonging to the light region in actuality should have been labeled as the background. This number of erroneously labeled points grew from frame to frame, as more and more points from the background region fit the statistics of what was erroneously labeled as the light region. In Method XV, we omit the position, motion, and Texture information all together, and make use of only the color feature. This indicates that combining the appropriate features usually outperforms the use of any single feature by itself.

In Table 5.5 and 5.6 we examine the performance of Methods XIV, and XVI respectively on a region by region basis over all 10 frames. This means that even though both of these two methods received overall performance ratings of 88.4% and 89.4% respectively, some of the regions were more accurately segmented than others. Also the size of the regions varied drastically. Region 6, the background/ upper

Method	% Labeled	% Correct	Method	% Labeled	% Correct
I	100	87.348	X	100	80.558
II	100	86.134	XI	100	77.540
III	100	87.443	XII	100	86.441
IV	100	86.243	XIII	100	88.358
V	100	84.099	XIV	100	88.414
VI	100	88.313	XV	100	78.931
VII	100	88.378	XVI	100	89.410
VIII	100	88.129	XVII	100	84.878
IX	100	87.371	XVIII	100	82.314

Table 5.4: Overall Performance Statistics for GYM Sequence Frames 1-10, see Table 5.3 for description of each experiment method.

audience comprised of more than half the scene. Regions 4 and 5, the mats and lower audience, comprised of about an eighth each. Region 3, the gymnasts in the background, comprised of about a tenth. Regions 2 and 7, the balance beam and the gymnast in the foreground, comprised of just over and under 5%. And finally the light, which is only present in the first 3-4 frames, comprised of 0.1% of the entire sequence.

Figures 5-8 and 5-9 show select frames of the mask file segmentation, and corresponding error signal of Methods XIV and XVI respectively. Figures 5-10 and 5-11 show similar segmentation results, with the only exception that a higher certainty threshold (of 0.09) was used, thus not all the pixels were labeled. Even though Method XIV has a lower objective performance rating, it could be argued that it has a better subjective performance. Specifically, both objectively and by looking at the mask segmentation, Method XIV does a better job of capturing the gymnast in the foreground, Region 7. Note that Method XVI does not use motion information, and that Method XIV uses a more robust PDF estimation process, namely, EM algorithm with deterministic annealing.

In addition to comparing the performance on a region by region basis, as in Tables 5.5 and 5.6 we can also compare the performance on a frame by frame basis. Table 5.7 shows the frame by frame performance over all 7 regions for the segmen-

Labeled Region	Ground Truth Region						
	1	2	3	4	5	6	7
1	51.23	0.00	0.00	0.52	0.00	0.11	0.00
2	0.39	89.23	2.32	0.80	1.48	0.19	4.99
3	15.83	6.86	78.49	11.61	9.23	0.52	1.72
4	3.94	1.11	14.75	84.60	11.83	0.04	0.00
5	17.38	0.54	4.42	2.47	70.15	1.39	2.06
6	11.24	0.01	0.01	0.00	6.05	96.21	1.89
7	0.00	2.26	0.01	0.00	1.27	1.53	89.33
Total Points	1548	43931	80038	107798	115456	423733	34856

Table 5.5: Each entry (r, c) (corresponding to row r, column c) in the table shows the percentage of pixels (over total of frames 1-10 of GYM sequence) that were labeled as region r, and actually belong to region c, based on **Method XIV**. The actual total raw number of points of each region is also shown, where the image size is 232 rows by 348 columns by 10 frames = 807360 total points.

Labeled Region	Ground Truth Region						
	1	2	3	4	5	6	7
1	83.59	0.19	0.03	0.60	0.00	0.71	0.00
2	1.10	94.71	5.66	0.32	3.23	0.10	4.53
3	4.13	1.25	72.25	12.65	7.52	0.03	0.01
4	1.10	0.15	11.92	84.68	5.22	0.12	0.00
5	2.58	1.78	9.74	1.45	74.50	0.33	0.22
6	7.49	0.12	0.21	0.26	8.55	97.52	7.38
7	0.00	1.81	0.18	0.05	0.99	1.20	87.87

Table 5.6: Each entry (r, c) (corresponding to row r, column c) in the table shows the percentage of pixels (over total of frames 1-10 of GYM sequence) that were labeled as region r, and actually belong to region c, based on **Method XVI**.

Method XIV			Method XVI		
Frame	**% Labeled**	**% Correct**	**Frame**	**% Labeled**	**% Correct**
1	100	91.430	1	100	91.690
2	100	92.385	2	100	91.545
3	100	88.997	3	100	91.164
4	100	91.704	4	100	89.965
5	100	90.907	5	100	91.903
6	100	87.912	6	100	88.710
7	100	85.054	7	100	89.005
8	100	85.990	8	100	87.024
9	100	84.976	9	100	87.585
10	100	84.781	10	100	85.510
Total	100	88.414	**Total**	100	89.410

Table 5.7: Frame by frame performance comparison of two methods of GYM sequence.

tation of each Method XIV and XVI. Recall from Table 5.1 that the training and tracking points were not at 100%. Notice in the accuracy of the training data, that even though the percentage in frame 3 is at 95% and in frame 6 it drops to 89%, the segmentation accuracy of Method XIV remains at almost 89% and 88% for the two frames, respectively. This exemplifies how incorrect points in the training data does not necessarily create a large error in the segmentation. Although by frames 9 and 10, the tracking starts to diminish more drastically, and the segmentation error follows.

Table 5.8 shows how the segmentation results vary as the certainty threshold is increased (from 0.0 to 0.25) for segmentation Methods XIV and XVI. A greater certainty threshold value signifies that we wish to retain those labels with a certainty greater than the threshold value, which means we are stricter about which points are labeled. That is we want to make sure that points with a very low certainty are left unlabeled. It follows that as the certainty threshold value increases, the percentage of points labeled decreases, but the percentage of accurately labeled points (with respect to the labeled points only) increases.

The certainty threshold control option is useful for a post-processing step that might use a labeling decision criteria other than the MAP rule we use. For example

Method XIV			Method XVI		
Cert Thresh	% Labeled	% Correct of labeled	Cert Thresh	% Labeled	% Correct of labeled
0.00	100.00	88.414	0.00	100.00	89.410
0.05	88.178	93.326	0.05	90.143	93.385
0.08	81.480	95.370	0.08	84.328	95.174
0.09	79.306	95.912	0.09	82.385	95.652
0.10	77.163	96.388	0.10	80.412	96.075
0.11	75.048	96.787	0.11	78.456	96.455
0.12	73.035	97.121	0.12	76.544	96.766
0.14	69.034	97.682	0.14	72.991	97.246
0.16	65.198	98.077	0.16	69.493	97.614
0.18	61.538	98.351	0.18	65.848	97.891
0.20	58.058	98.518	0.20	61.785	98.087
0.22	54.601	98.629	0.22	57.645	98.219
0.25	49.352	98.714	0.25	51.123	98.417

Table 5.8: Performance analysis of two segmentation methods of GYM sequence, frames 1-10, as certainty threshold is varied.

a K-nearest neighbor (or KNN) algorithm looks at the label values around the neighborhood of an unclassified sample, and assigns a label value to the class with the most number of neighbors in that region. Variations of KNN involve whether to keep the neighborhood size constant, or to keep constant the required number of "votes" needed to label an unlabeled sample. The declassification and neighbor fill programs procedures are discussed in more detail in Section 4.5.

Table 5.9 shows the performance by region of the KNN post processing step applied to the segmentation result of experiment Method XIV with a certainty threshold of 0.09. So after the KNN procedure, the final segmentation contains 100% of the points labeled. The table included shows the results on a region basis (compare to Table 5.5. Over all 10 frames and all the regions, the total accuracy was at 90.087% compared to 88.414% for experiment XIV directly (see Table 5.4). The frame by frame analysis for this KNN post-processing method is not shown, but it is roughly 2% better than the method XIV with no certainty threshold (see Table 5.7).

Figures 5-9 through 5-12 show selected frames of the segmentation as well as the

Labeled Region	Ground Truth Region						
	1	2	3	4	5	6	7
1	54.97	0.00	0.00	0.44	0.00	0.21	0.00
2	0.58	94.57	2.45	0.89	2.14	0.32	3.65
3	12.47	2.79	82.42	9.29	8.66	0.47	0.82
4	4.07	0.40	10.63	87.23	11.39	0.00	0.00
5	7.69	0.54	4.38	2.16	70.52	0.46	0.83
6	20.22	0.05	0.13	0.00	5.97	97.13	3.12
7	0.00	1.65	0.00	0.00	1.31	1.41	91.59

Table 5.9: Each entry (r, c) (corresponding to row r, column c) in the table shows the percentage of pixels (over total of frames 1-10 of GYM sequence) that were labeled as region r, and actually belong to region c, based on K-nearest neighbor algorithm applied to mask segmentation output using **Method XVI** and certainty threshold of 0.09.

error of methods whose results have been discussed objectively. We include the images themselves since the "subjective" evaluation is often different than the objective measurements included in the tables above. Ideally we would have like to have shown all the frames displayed as a video sequence, of course that is not possible on print medium. Instead, we have selected several frames of some of the experiments.

Figure 5-8 shows the segmentation mask file and error of experiment XIV with no certainty threshold applied, *i.e.* all points labeled. Figure 5-9 shows the segmentation mask file and error of experiment XVI with no certainty threshold applied. Figure 5-10 shows the segmentation mask file and error of experiment XIV with a certainty threshold value of 0.09, corresponding to about 79.3% labeled. Figure 5-11 shows the segmentation mask file and error of experiment XVI with a certainty threshold value of 0.09, corresponding to about 82.4% labeled. Figure 5-12 shows the segmentation mask file and error resulting from the KNN post-process applied to experiment XIV with a certainty threshold value of 0.09. Thus 100% of the points were labeled. 79.3% (corresponding to the points with the highest certainty) using the MAP criterion, and the remaining 20.7% using the K-nearest neighbor algorithm.

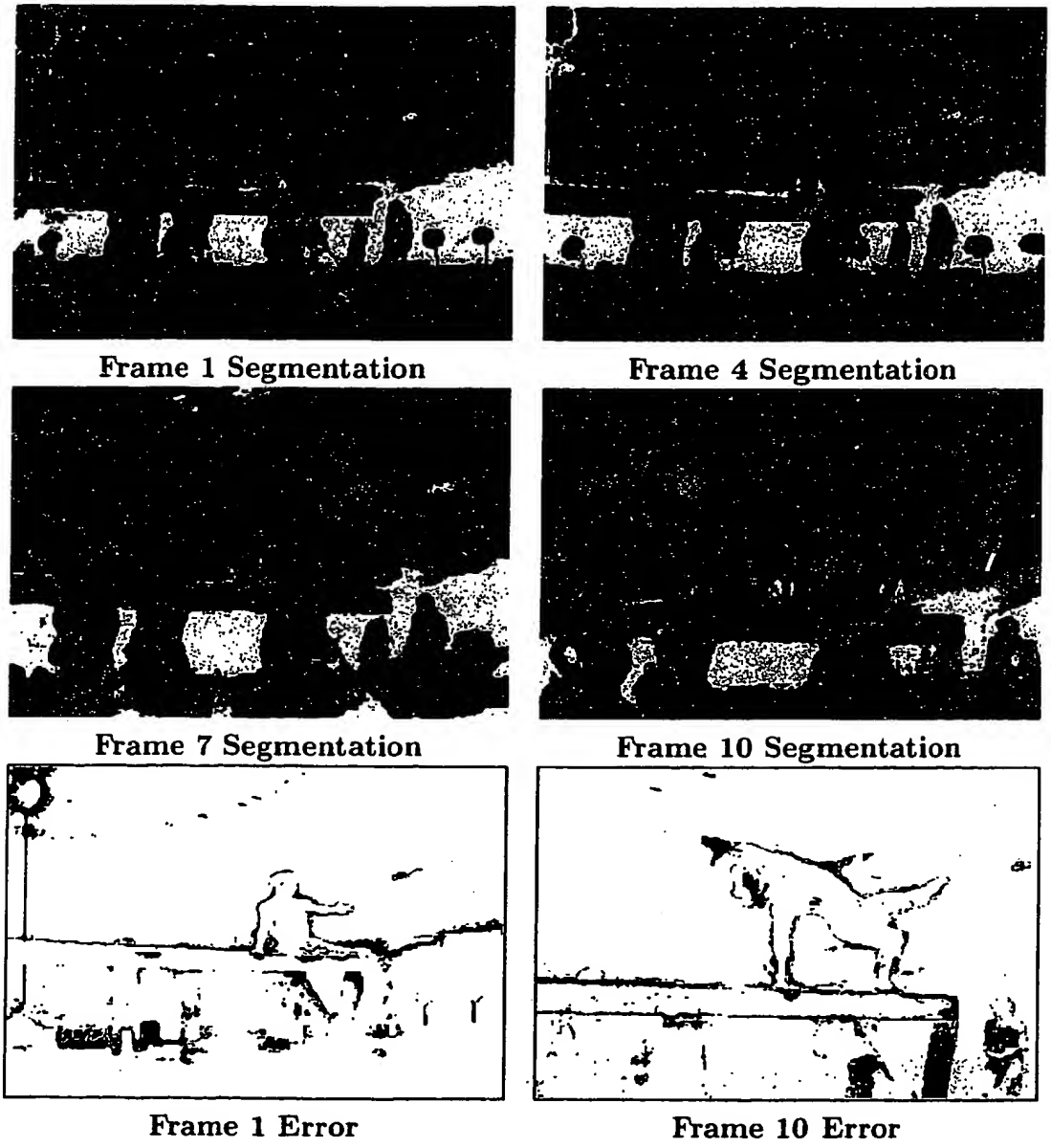


Figure 5-8: Selected frames of segmentation of GYM and error signal, using Method XIV

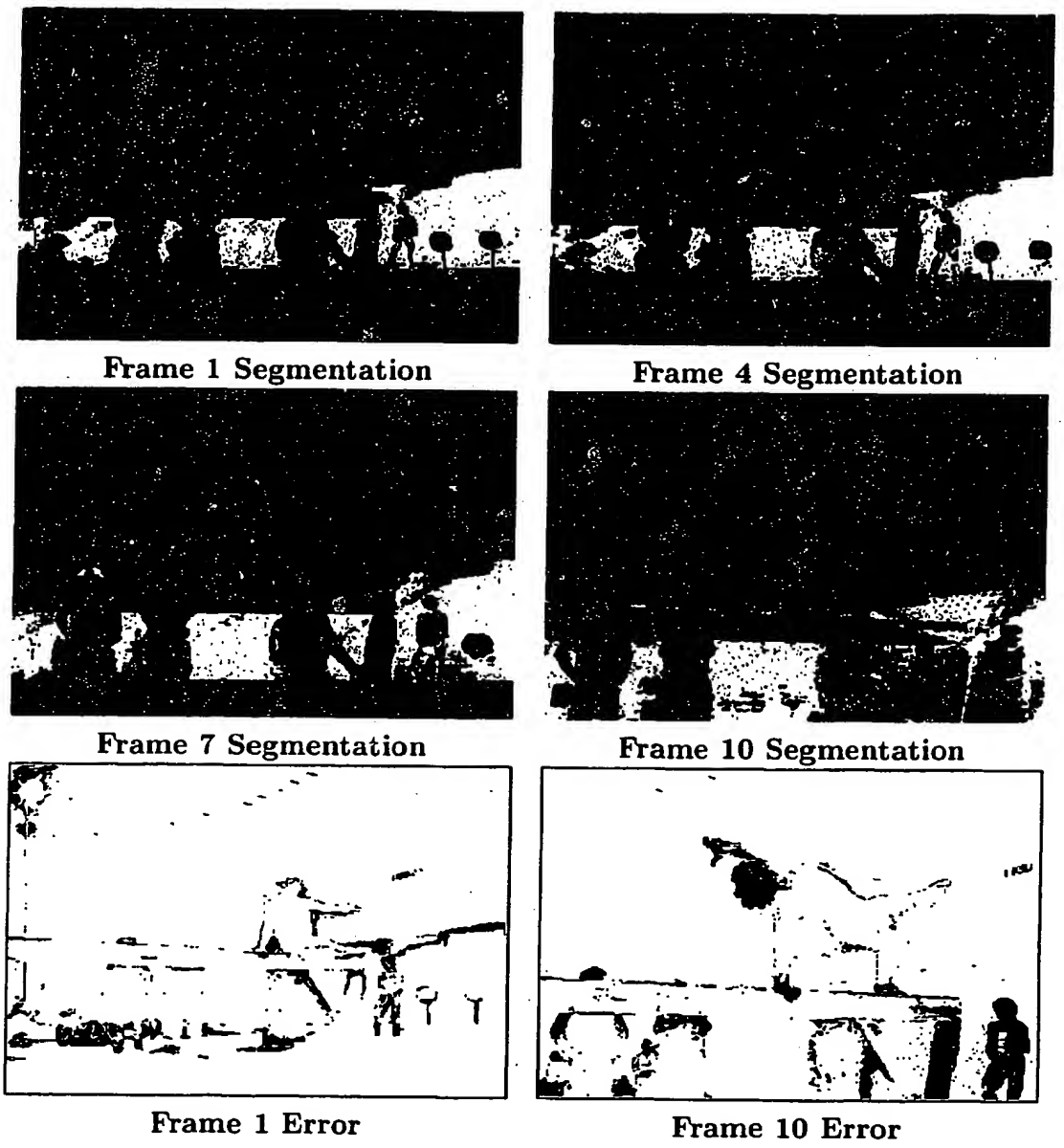


Figure 5-9: Selected frames of segmentation of GYM and error signal, using Method XVI

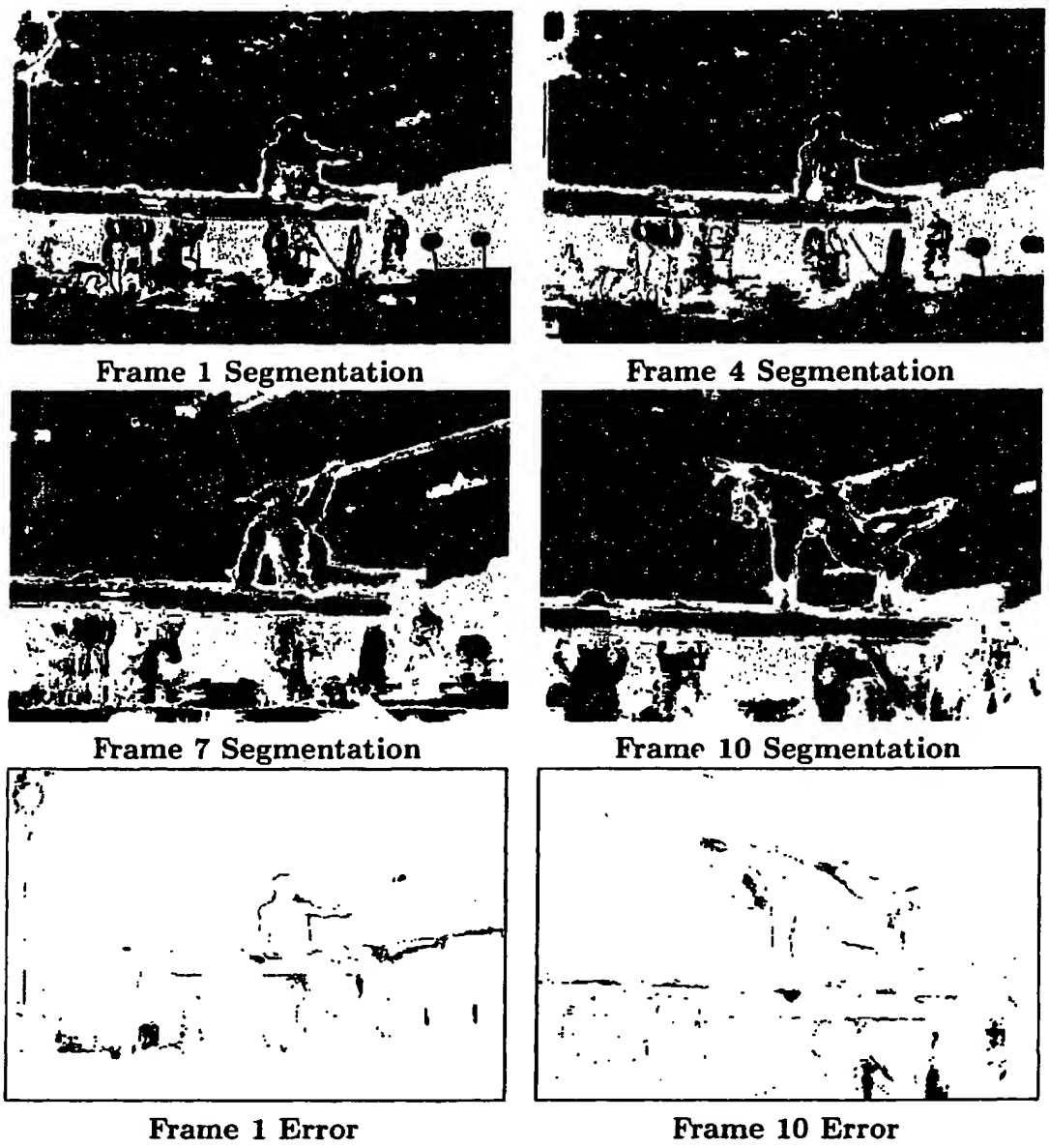


Figure 5-10: Selected frames of segmentation of GYM and error signal, using Method XIV and certainty threshold of 0.09

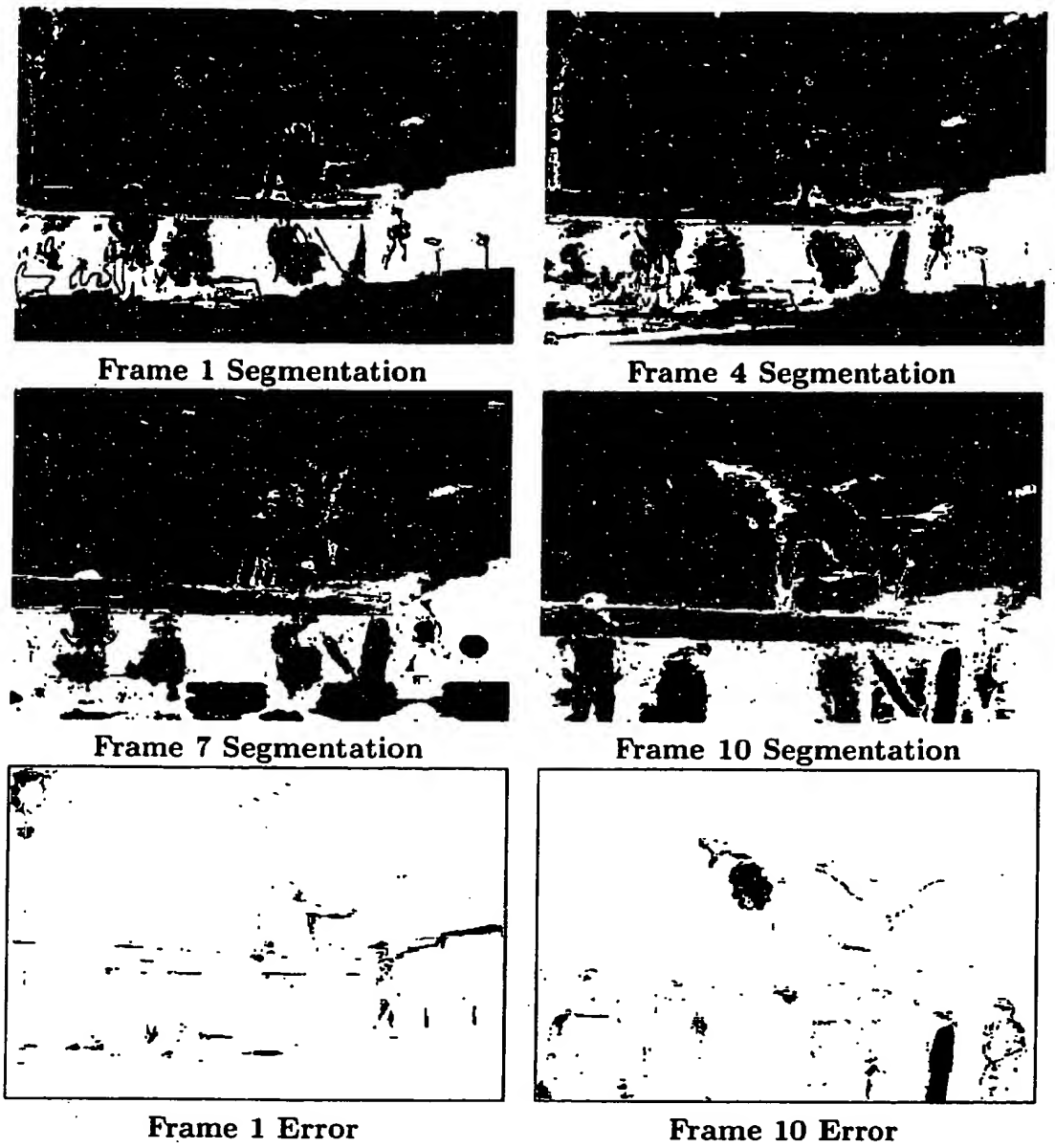


Figure 5-11: Selected frames of segmentation of GYM and error signal, using Method XVI and certainty threshold of 0.09

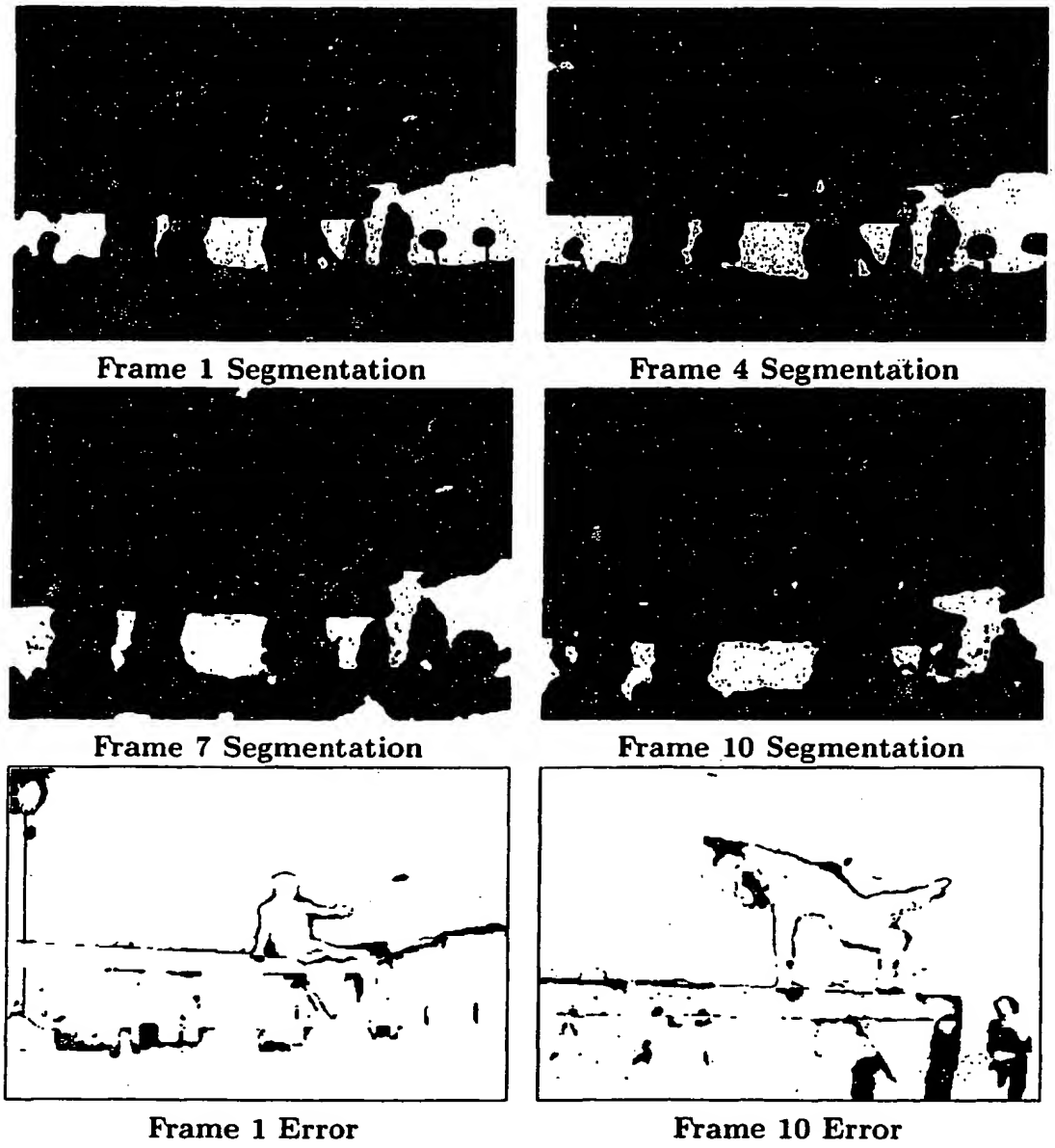


Figure 5-12: Selected frames of segmentation of GYM and error signal, using the KNN algorithm on the mask file shown in Figure 5-10.

5.2.2 GMV Sequence

As with the GYM sequence, in the Good Morning Vietnam (GMV) sequence, we will show selected frames of the original, manual segmentation, tracking and a couple of variations of segmentation experiments. We will again display tables of "objective" comparisons of different methods. Figure 5-13 shows selected frames of the original segmentation, as well as the manual segmentation. Notice how there is less motion in the GMV sequence compared to the GYM sequence.

Figure 5-14 shows selected frames of the tracking points of GMV. The first frame are the points supplied directly by the user. The remaining frames are the result of the tracking program. At each frame the tracked points are used as "training" to estimate the PDF of each feature at each region. In the GMV sequence the accuracy of the training and tracking points was about 99.5% accurate relative to the ground truth, which is much higher than the GYM sequence where the accuracy of the tracking points fell very quickly from frame to frame. Of the 3047 samples provided as training points in the first frame, there were only about 2300 points remaining in each successive frame. The lost points are a result of image boundary artifacts caused by the tracking process. Specifically, the size of the image boundary artifacts are function of the search range and block size.

Table 5.10 shows the objective segmentation results of about 14 of the 18 experiments listed in 5.3. The ones omitted were either because the performance was obviously going to be low, or because they would be too similar to an experiment already described. In fact, of the14 experiment results shown, the lowest accuracy is at 85.3%, and the highest at 89.8%, which shows the tightness of the results. Notice that even though GMV has less motion and "better" training data compared to GYM, its segmentation results are surprisingly about par with the results from GYM. A look closer at the results on a region by region analysis make give a better understanding.

Tables 5.11 and 5.12 show the segmentation results over all ten frames of GMV sequence, on a region by region performance, for methods VII and XIII, respectively. Both results perform qualitatively similar, about 95% in regions 1, 3 and 4 (the

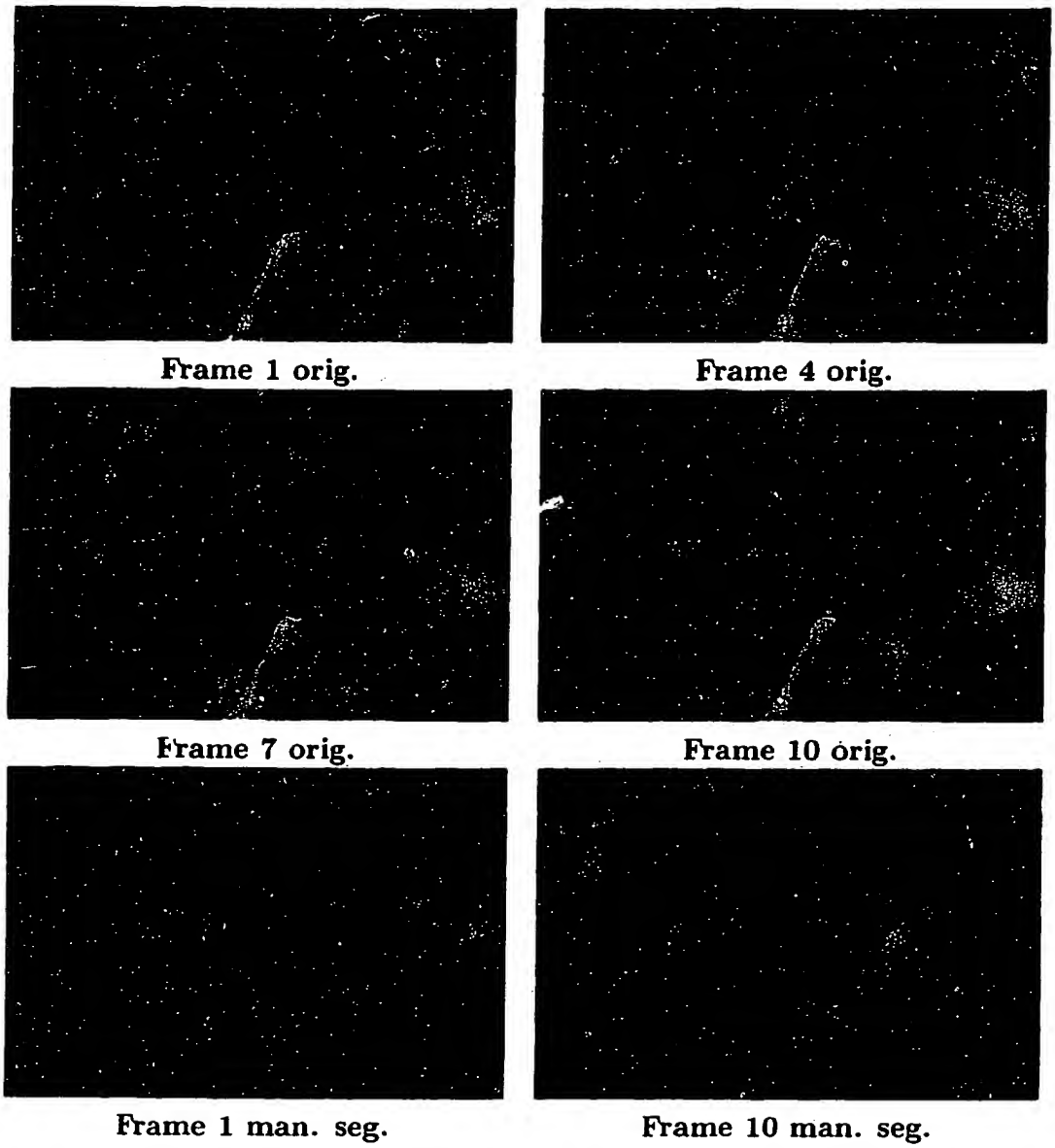


Figure 5-13: Original and manually segmented frames from GMV sequence.

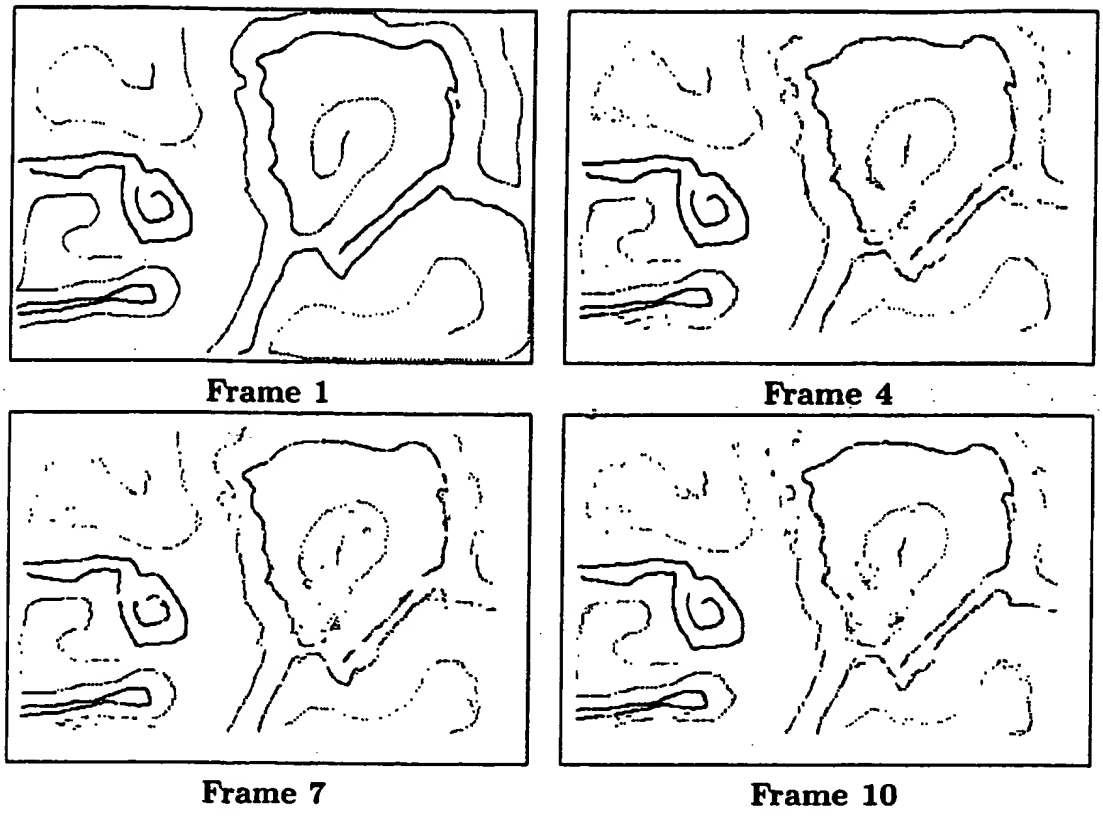


Figure 5-14: Training Data points in Frame 1 of GMV, and tracked location of those points at Frames 4,7 and 10.

Method	% Labeled	% Correct	Method	% Labeled	% Correct
I	100	87.893	IX	100	86.179
II	100	86.371	X	100	88.483
III	100	88.873	XI	100	85.331
IV	100	86.633	XII	100	88.129
VI	100	88.936	XIII	100	89.853
VII	100	89.351	XIV	100	89.049
VIII	100	88.543	XVI	100	89.215

Table 5.10: Overall Performance Statistics for GMV Sequence Frames 1-10, see Table 5.3 for description of each experiment method.

Labeled Region	Ground Truth Region			
	1	2	3	4
1	96.31	6.21	0.30	0.00
2	3.17	84.66	2.50	4.32
3	0.05	1.60	93.56	0.59
4	0.47	7.53	3.65	95.09
Total Points	59446	456524	174484	173546

Table 5.11: Each entry (r, c) (corresponding to row r, column c) in the table shows the percentage of pixels (over total of frames 1-10 of GMV sequence) that were labeled as region r, and actually belong to region c, based on **Method VII**. The actual total raw number of points of each region is also shown, where the image size is 240 rows by 360 columns by 10 frames = 864000 total points.

microphone, shirt, and face regions) but low to mid 80 percentile in region 2 (the background). Method XIII performs slightly better in regions 1,3, and 4 and slightly worse in region 2. Compared to the GYM sequence, the total performance is about equal. The difference is the performance on a region by region basis. The GYM sequence performed well in detecting its background region and not as well in detecting some of the other regions. The reverse is true of the GMV sequence. The background in GMV is difficult to detect, because it has one or two sections, the horizontal line for example, that are different from the rest of the background section, and at the same time, actually quite similar to the statistics of one of the other regions. The fact that a small portion of the background image is different from the rest of the background means that in the PDF estimation the statistics that represent that small portion don't generate a high enough likelihood. By itself, that would not be so bad, but when added in conjunction that another region has similar statistics to the small portion of the background, then selecting the other region will usually "win" over selecting the background.

Table 5.13 shows the frame by frame analysis of the performance of methods VII and XIII over all regions. Since motion is not much of a factor, and the integrity of the training/tracking points remain intact, the performance is about the same over

Labeled Region	Ground Truth Region			
	1	2	3	4
1	97.43	7.34	0.00	0.00
2	2.56	83.56	1.90	1.25
3	0.00	1.89	95.23	0.34
4	0.00	7.20	2.87	98.41

Table 5.12: Each entry (r, c) (corresponding to row r, column c) in the table shows the percentage of pixels (over total of frames 1-10 of GMV sequence) that were labeled as region r, and actually belong to region c, based on **Method XIII**.

Method VII			Method XIII		
Frame	% Labeled	% Correct	Frame	% Labeled	% Correct
1	100	89.054	1	100	90.641
2	100	89.142	2	100	90.571
3	100	89.728	3	100	90.205
4	100	89.347	4	100	88.700
5	100	89.141	5	100	89.679
6	100	89.044	6	100	89.862
7	100	89.126	7	100	89.229
8	100	89.405	8	100	90.046
9	100	89.568	9	100	89.764
10	100	89.955	10	100	89.833
Total	100	89.351	**Total**	100	89.853

Table 5.13: Frame by frame performance comparison of two methods of GMV sequence.

all 10 frames. In fact, the performance would continue to remain the same for frames even past the first 10, until major motion or change is introduced to the sequence. Manual segmentation data was only made available for the first 10 frames, which is why we stop the analysis there too.

Table 5.14 shows how the performance increases (relative to the labeled points) as the certainty threshold increases. Qualitatively these results are very similar to GYM sequence, and if one were to plot performance as a function of percentage of points labeled, the shapes of the plots for any of the GYM experiments would be very similar to the shapes of the plots for any of the GMV sequences.

Method VII			Method XIII		
Cert Thresh	% Labeled	% Correct of labeled	Cert Thresh	% Labeled	% Correct of labeled
0.00	100.00	89.351	0.00	100.00	89.853
0.05	88.807	93.297	0.05	94.784	91.972
0.08	80.866	94.871	0.08	91.372	93.139
0.09	77.932	95.310	0.09	90.196	93.501
0.10	74.885	95.704	0.10	88.992	93.855
0.11	71.815	96.055	0.11	87.766	94.196
0.12	68.785	96.386	0.12	86.492	94.549
0.14	62.718	96.953	0.14	83.821	95.219
0.16	56.634	97.406	0.16	80.909	95.885
0.18	50.346	97.714	0.18	77.787	96.491
0.20	43.818	97.934	0.20	74.361	97.081
0.22	37.242	98.114	0.22	70.733	97.641
0.25	27.335	98.288	0.25	64.679	98.505

Table 5.14: Performance analysis of two segmentation methods of GMV sequence, frames 1-10, as certainty threshold is varied.

Figures 5-15 through 5-18 show selected frames of methods VII and XIII each using a certainty threshold of 0.0 (i.e. 100% of the points labeled) and 0.09. Interestingly, with the same certainty threshold provided for both Methods VII and XIII, they produce quite a difference with regard to the percentage of samples labeled. Method VII has just under 78% labeled, and Method XIII has just over 90% labeled. In fact from Table 5.14, that trend continues. Which implies that overall, method XIII not only performs slightly better than Method VII, but it also has a higher certainty.

We omitted displaying the result of the neighbor fill program (see Figure 5-12 and Table 5.9) in the GMV sequence because it only gave about a 0.5% advantage over all 10 frames, when comparing Method XIII certainty threshold 0.00 to the result of the KNN algorithm on Method XIII certainty threshold 0.09.

5.2.3 FISH Sequence

The last of our three test image sequences comes from A Fish Called Wanda, (FISH). The camera pans left to follow the actors in the foreground, keeping them in the

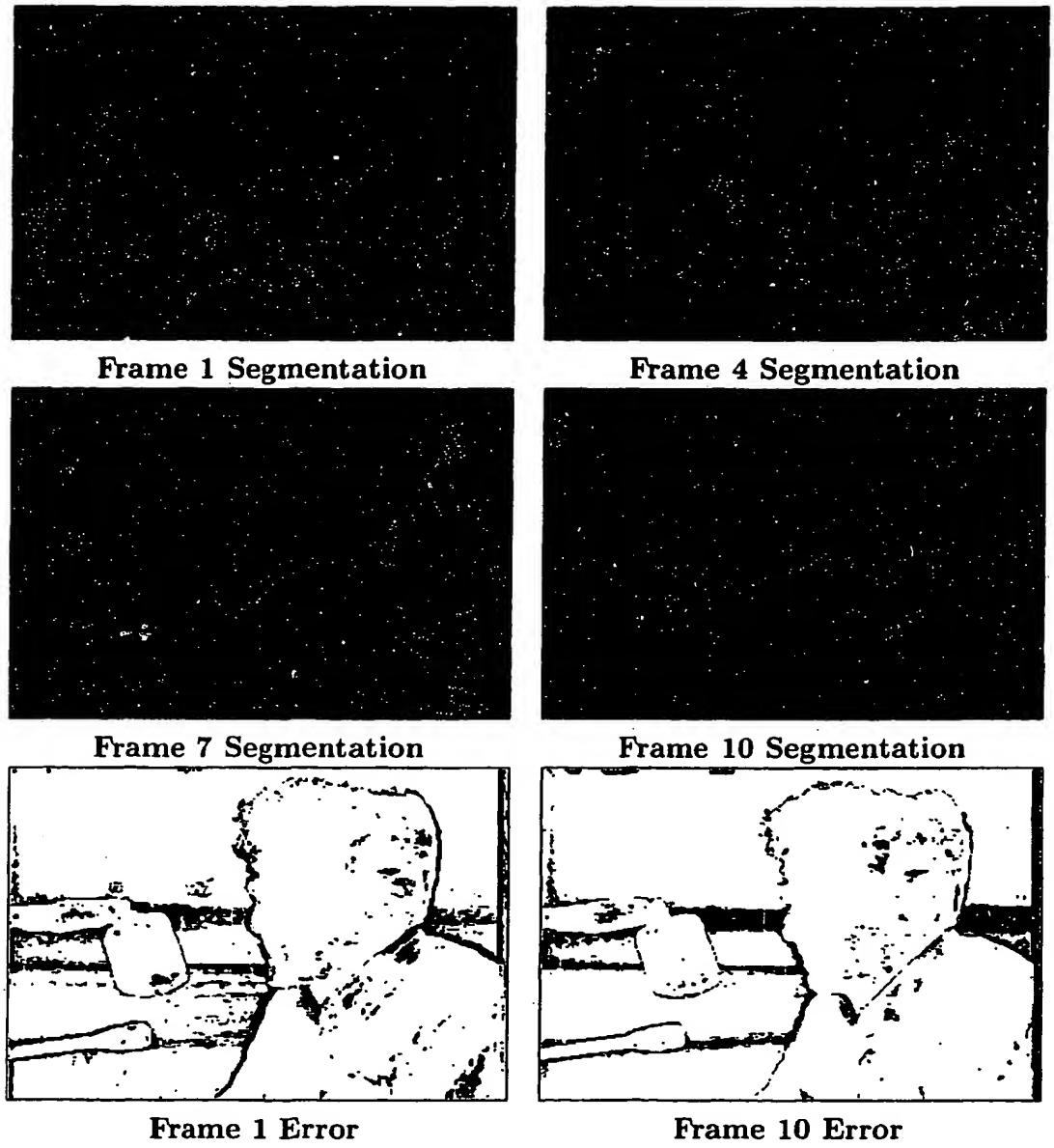


Figure 5-15: Selected frames of segmentation of GMV and error signal, using Method VII

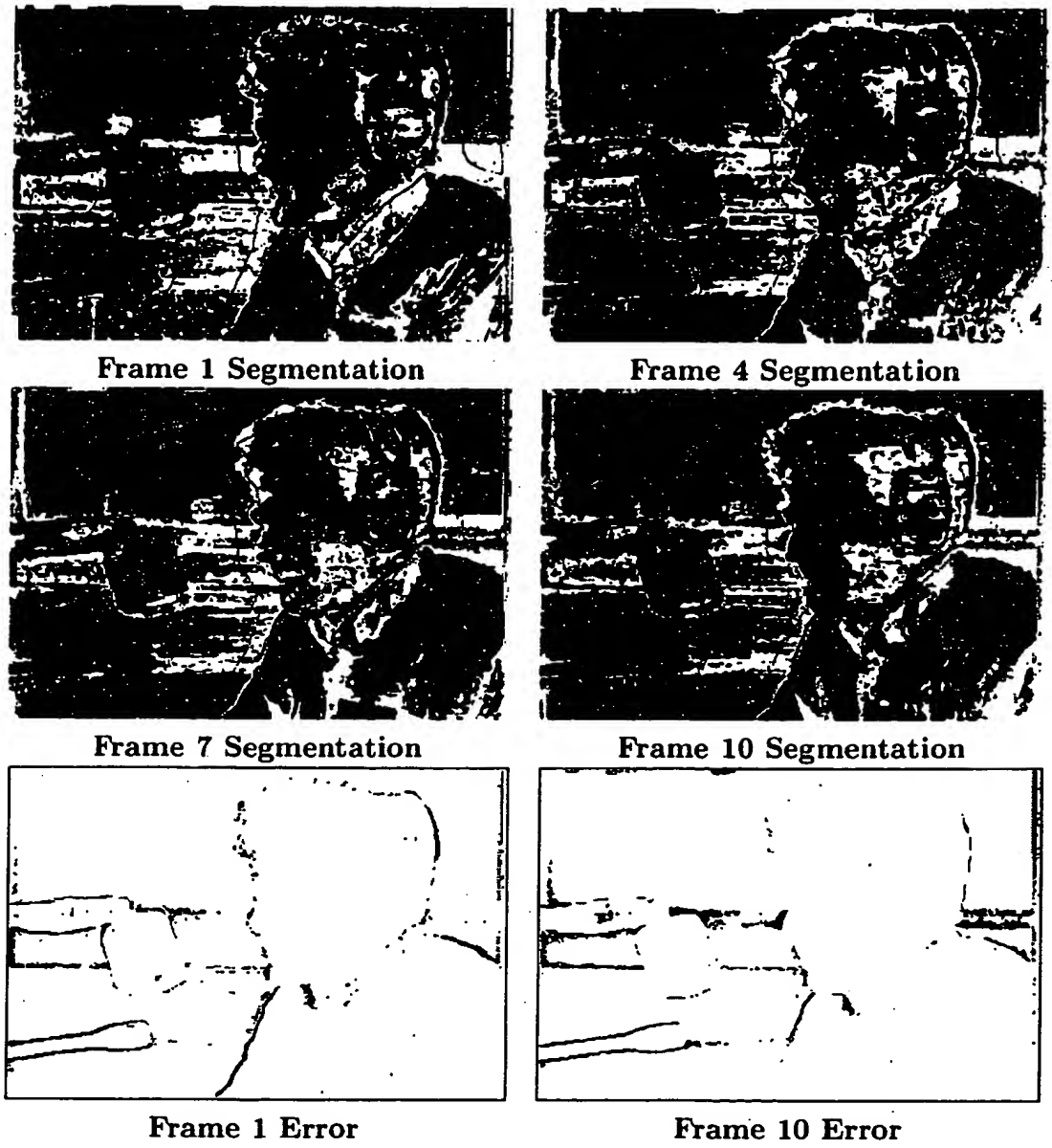


Figure 5-16: Selected frames of segmentation of GMV and error signal, using Method VII and certainty threshold of 0.09

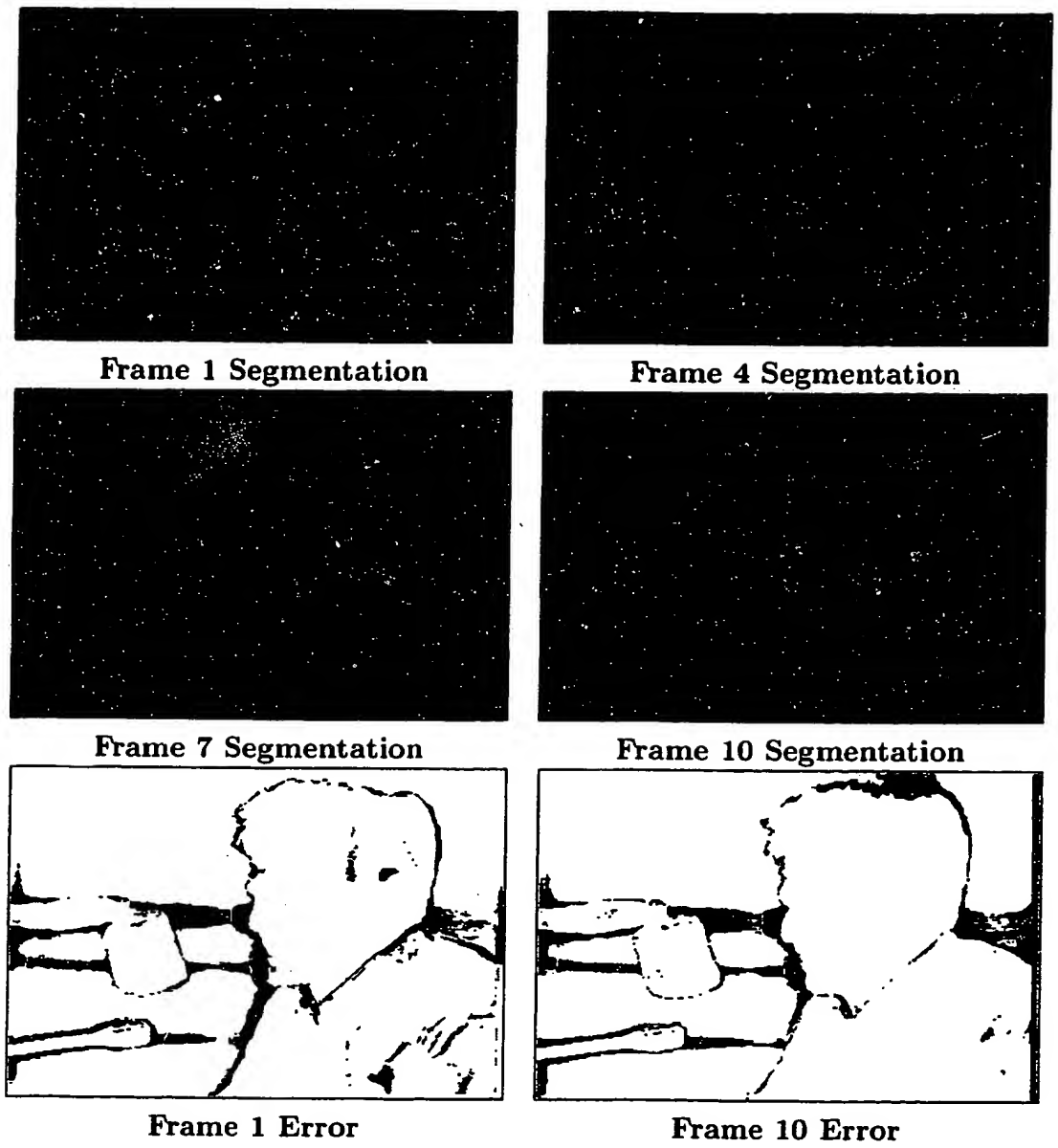


Figure 5-17: Selected frames of segmentation of GMV and error signal, using Method XIII

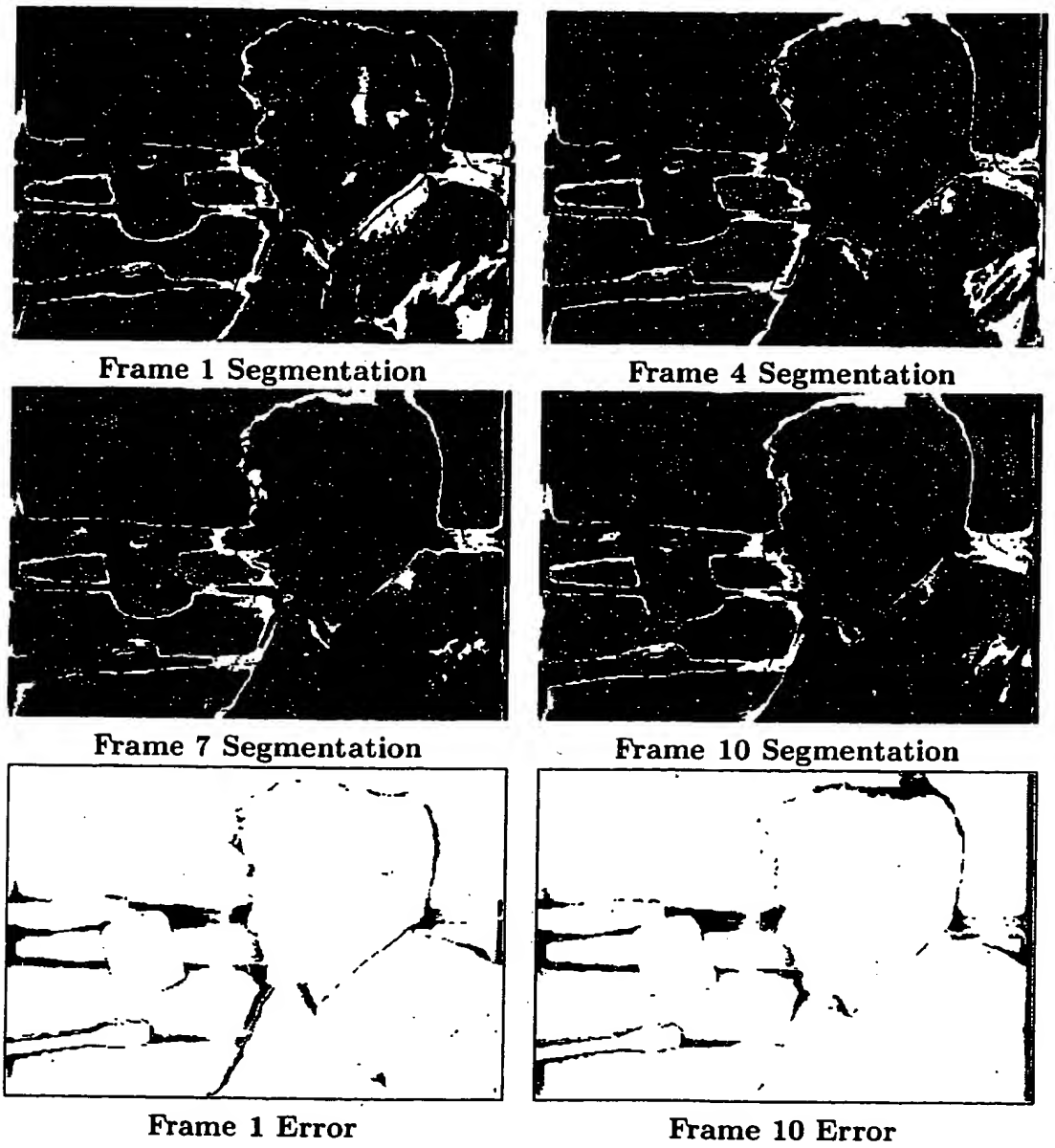


Figure 5-18: Selected frames of segmentation of GMV and error signal, using Method XIII and certainty threshold of 0.09

Method	% Labeled	% Correct	Method	% Labeled	% Correct
I	100	86.296	VII	100	85.729
II	100	83.372	VIII	100	85.418
III	100	85.603	IX	100	82.481
IV	100	82.283	XII	100	85.193
VI	100	86.293			

Table 5.15: Overall Performance Statistics for FISH Sequence Frames 1-10, see Table 5.3 for description of each experiment method.

center of the frame. Figure 5-19 shows selected frames of the original sequence as well as from the manual segmentation mask file. The user has decided to label 6 regions: Region 1 is the actress on the left; Region 2 is the actor on the right; Region 3 corresponds to the two cars in the background; Region 4 is the sky (upper left portion); Region 5 is the buildings, and Region 6 is the ground.

Figure 5-20 shows selected frames resulting from the user training, and point tracking process. The user supplies 2581 training samples points in the first frame, of which 31 of them or 1.2% differ from the first frame of the manual segmentation mask file. The remaining nine frames contain an average of 1730 tracked training points per frame, of which about 73 or 4.2% of them differ from the ground truth.

Table 5.15 shows the list of 9 segmentation experiments performed on the FISH sequence. Again, we have shortened the list slightly, compared to the list of experiments performed on the GMV sequence shown in Table 5.10. We have omitted the variations on the tracking and PDF estimation algorithms completely. The remaining segmentation experiments seem to have similar enough characteristics.

Tables 5.16 and 5.17 focus in on Method I, showing a region by region analysis, and a frame by frame analysis, respectively. Note that Method I has about 86.3% accuracy or 13.7% error. If we take a deeper look at Table 5.16, we can calculate that about 43% of the errors is due to between regions 5 and 6, (26.8% region 6 labeled as 5, and 16.7% region 5 labeled as 6). Suppose that we combine these two regions *a posteriori*, our error would drop from 13.7% to about 7.8%. In other words, suppose we originally considered the image to have 5 regions, instead of 6, where regions 1-4

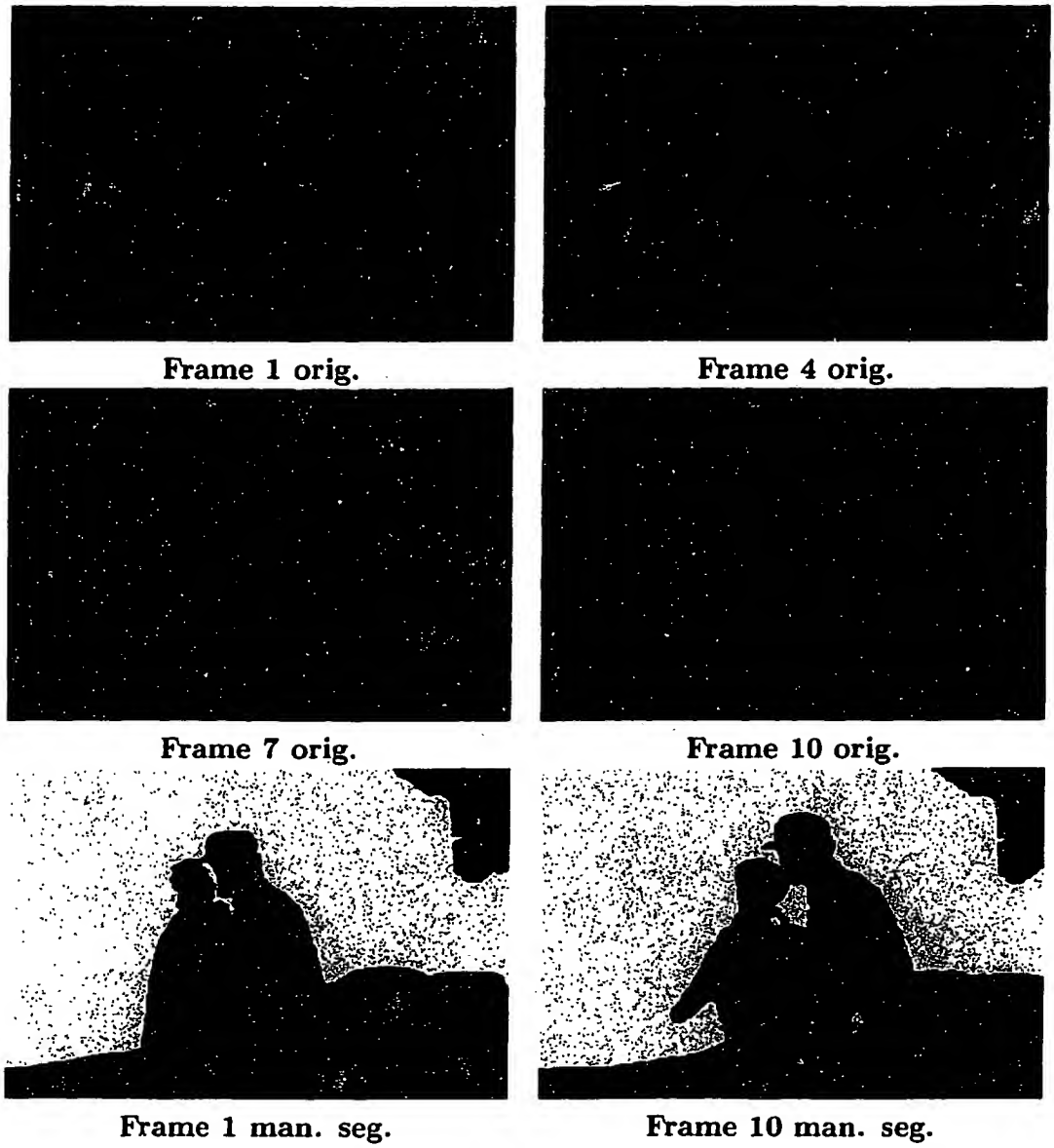


Figure 5-19: Original and manually segmented frames from FISH sequence.

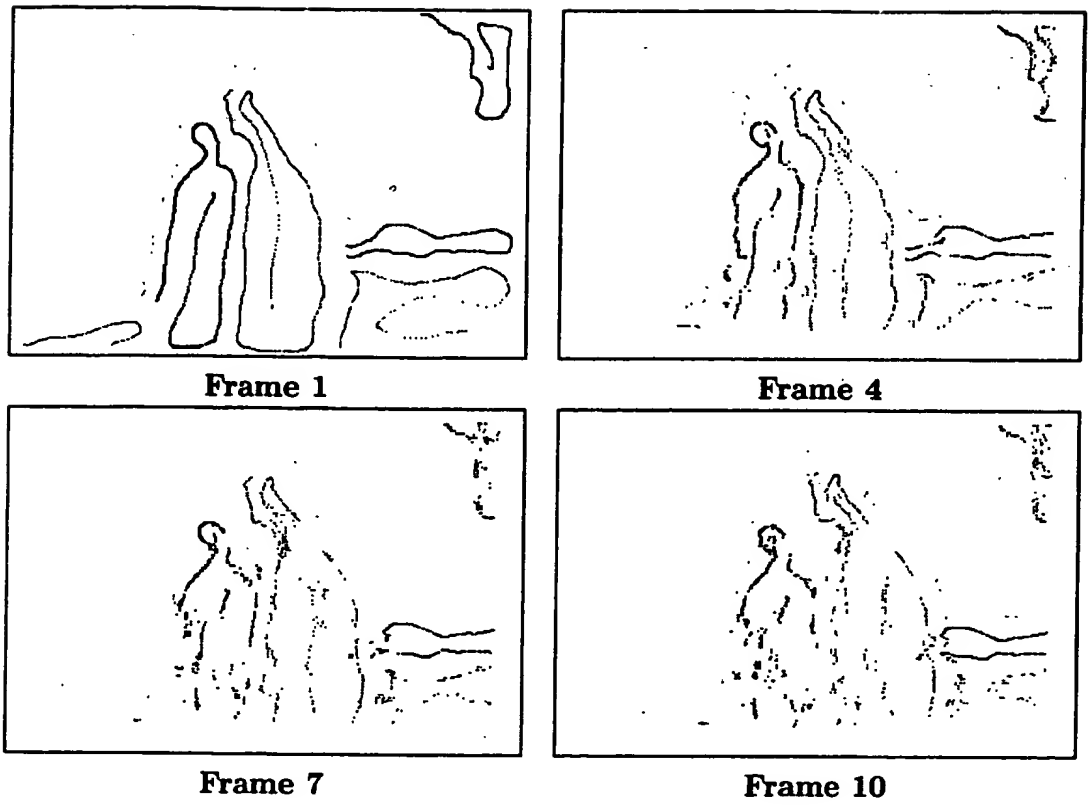


Figure 5-20: Training Data points in Frame 1 of FISH, and tracked location of those points at Frames 4,7 and 10.

Labeled Region	Ground Truth Region					
	1	2	3	4	5	6
1	83.05	2.06	0.01	0.64	5.02	1.36
2	8.84	92.45	1.24	0.73	0.81	1.56
3	0.05	0.43	90.33	0.01	0.17	1.95
4	0.00	0.00	0.00	83.95	0.12	0.00
5	7.75	4.98	8.35	14.33	89.76	28.44
6	0.31	0.08	0.07	0.33	4.12	66.69
Total Points	85272	126291	27572	35016	478356	111493

Table 5.16: Each entry (r, c) (corresponding to row r, column c) in the table shows the percentage of pixels (over total of frames 1-10 of FISH sequence) that were labeled as region r, and actually belong to region c, based on **Method I**. The actual total raw number of points of each region is also shown, where the image size is 240 rows by 360 columns by 10 frames = 864000 total points.

were the same, and the new 5th region were a union of regions 5 and 6. Then if we were to run the segmentation process using all 6 regions (using Method I for example), we would obtain the result described above. But as an additional step in the process, we would need to combine the segmentation of regions 5 and 6, in order to produce a final segmentation with 5 regions. In this manner, we would have an error of 7.8%

Table 5.18 shows how the accuracy increases (with respect to the number of labeled points) as the certainty threshold increases (or the percentage of labeled points decreases). This result is very similar to the results obtained with the GYM and GMV sequences shown in Tables 5.8, and 5.14.

Figures 5-21 and 5-22 show selected frames of segmentation experiment method I, with a certainty threshold of 0.00 and 0.09 respectively, along with the corresponding error signal for each segmentation result. Again, note that the error signal, corresponding to the segmentation using a certainty threshold of 0.09, is only with respect to the labeled samples. Thus an unlabeled sample does not show up in the error signal.

Method I		
Frame	**% Labeled**	**% Correct**
1	100	89.063
2	100	86.323
3	100	85.259
4	100	84.924
5	100	86.032
6	100	86.175
7	100	87.338
8	100	86.863
9	100	85.138
10	100	85.848
Total	100	86.296

Table 5.17: Frame by frame performance of FISH sequence.

Method I		
Cert Thresh	**% Labeled**	**% Correct of labeled**
0.00	100.00	86.296
0.05	90.614	90.294
0.08	84.998	92.519
0.09	83.060	93.229
0.10	81.072	93.894
0.11	79.033	94.518
0.12	76.953	95.096
0.14	72.501	96.133
0.16	67.560	97.014
0.18	62.301	97.739
0.20	56.719	98.381
0.22	50.991	98.841
0.25	42.641	99.216

Table 5.18: Performance analysis of two segmentation methods of FISH sequence, frames 1-10, as certainty threshold is varied.

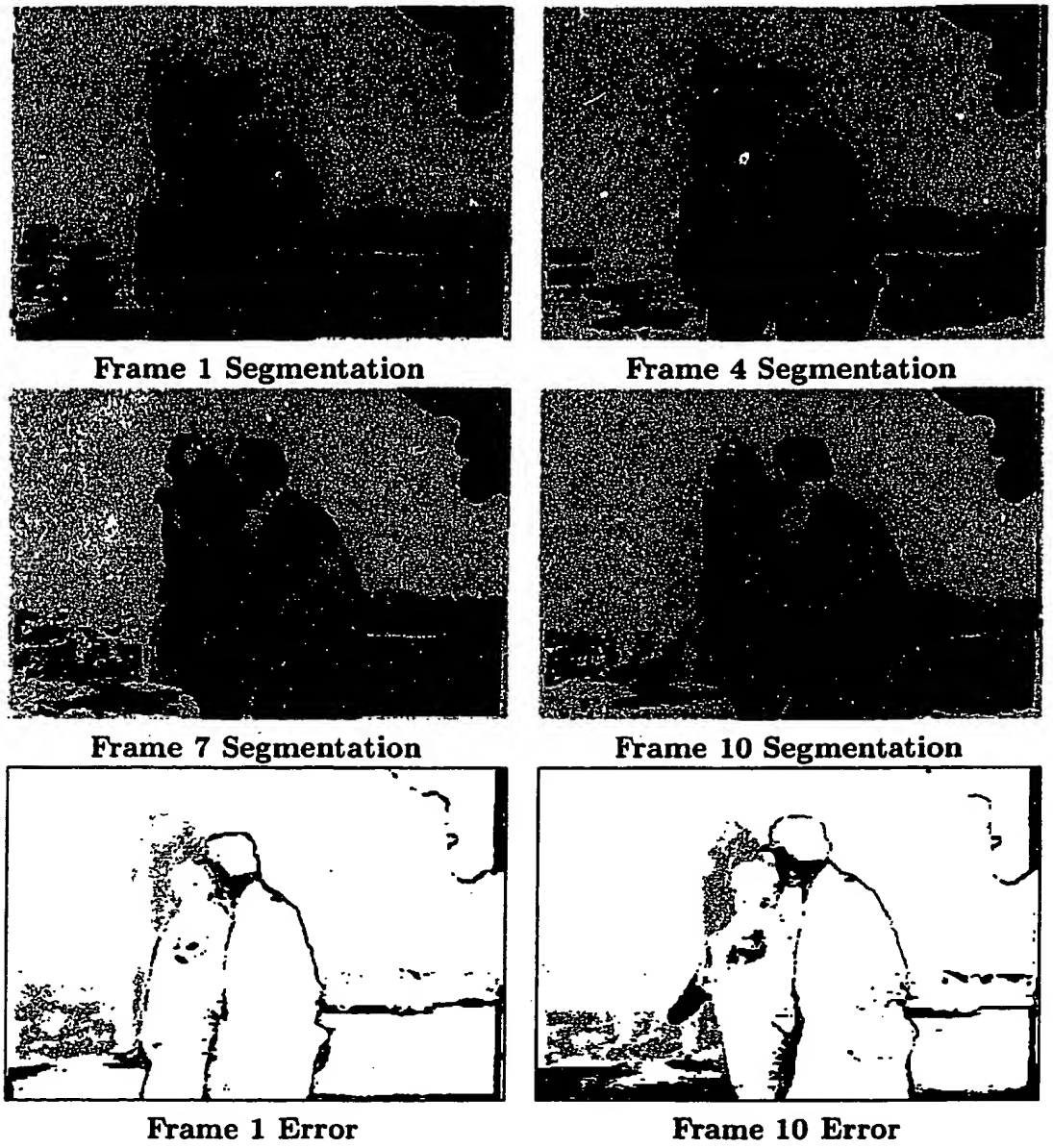


Figure 5-21: Selected frames of segmentation of FISH and error signal, using Method I

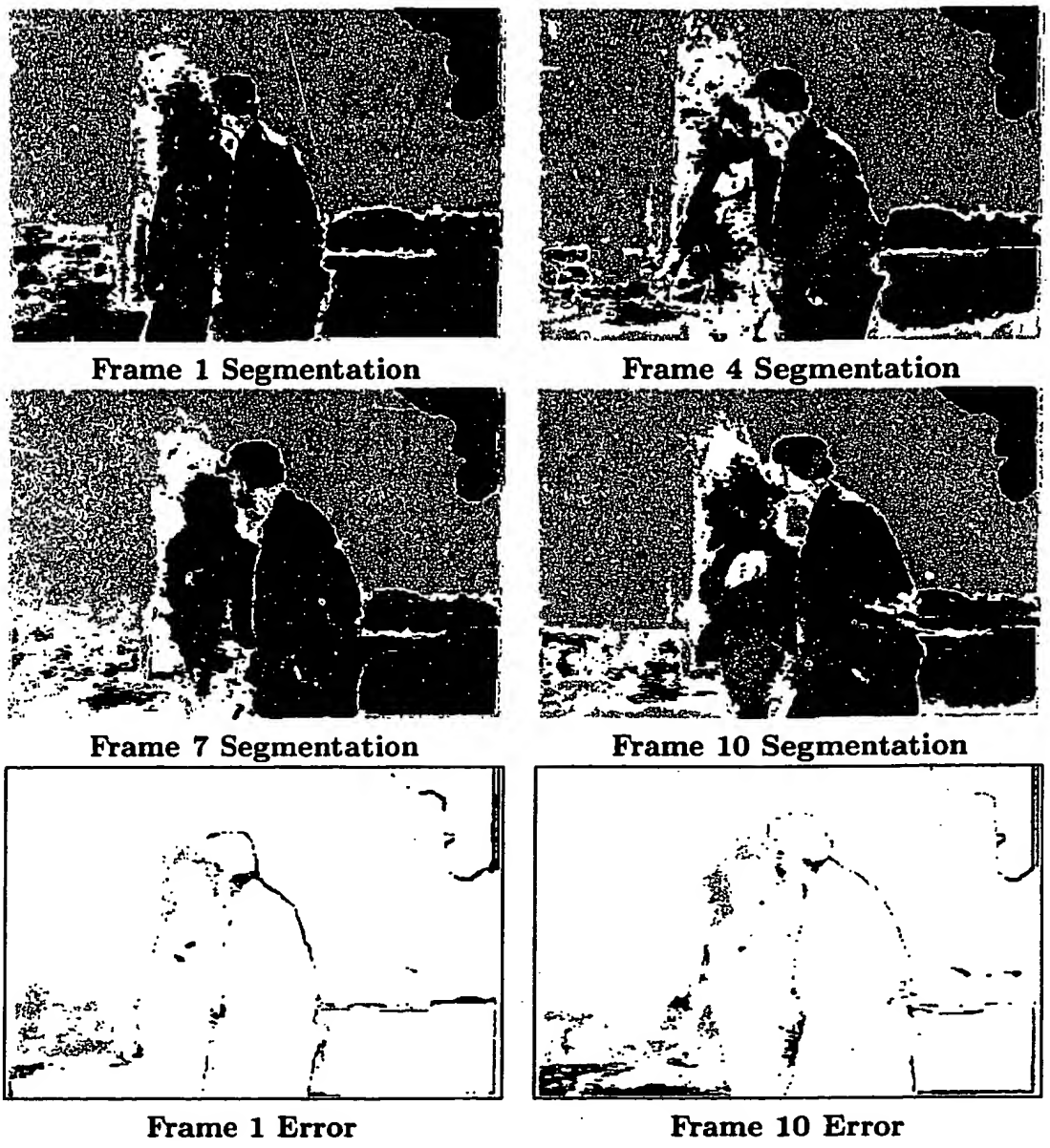


Figure 5-22: Selected frames of segmentation of FISH and error signal, using Method I and certainty threshold of 0.09

5.3 More Images

In this section we will show selected frames of original, tracking, and segmentation on image sequences for which no manual segmentation (or ground truth) was made available. The DANCE sequence is 59 frames, and shows 3 people dancing in front of a camera. The JEANINE sequence is 30 frames, it is a quasi-static image of a woman holding an advertisement box. And The SKIPPER sequence is 30 frames, and was shot on a boat so there is a lot of camera motion.

Figures 5-23, 5-24, and 5-25 show six frames of each of the DANCE, JEANINE, and SKIPPER sequence. Figure 5-26 shows the training of all three sequences, and also shows the TRACKING of selected frames from each of DANCE and SKIPPER. The tracking of the JEANINE sequence was not computed. Rather, the segmentation of JEANINE was done using the PDF estimate of the first frame only, as the basis for all remaining frames since the sequence is quasi-static. Therefore, no tracking algorithm was computed. However, a motion vector was computed as one of the features for the JEANINE sequence. For all three sequences, experiment method IX, as described in Table 5.3, was used. The only exception is the tracking method used for Jeanine was not method A, as described in Table 5.2, but instead no tracking was computed.

Figures 5-27 and 5-28 show the segmentation results of the DANCE sequence using experiment method IX, with a certainty threshold of 0.0 and 0.09, respectively. Since there is no ground truth or manual segmentation data available, there is no objective calculation that calculates the accuracy of these methods. We can calculate the percentage of labeled points as a function of certainty threshold. In Figure 5-28, the certainty threshold is 0.09, and the percentage of unlabeled points over all 59 frames is 19.0%, while over the first frame it is 14.1% unlabeled. Subjectively, we can note that the portion of the image corresponding to the ground does not segment out very well. The reason for this is due to the fact that there is no training data for the ground (since the tracking algorithm has edge boundary effects). Other factors, such as lighting, and the fact that the column position PDF is pretty tight around

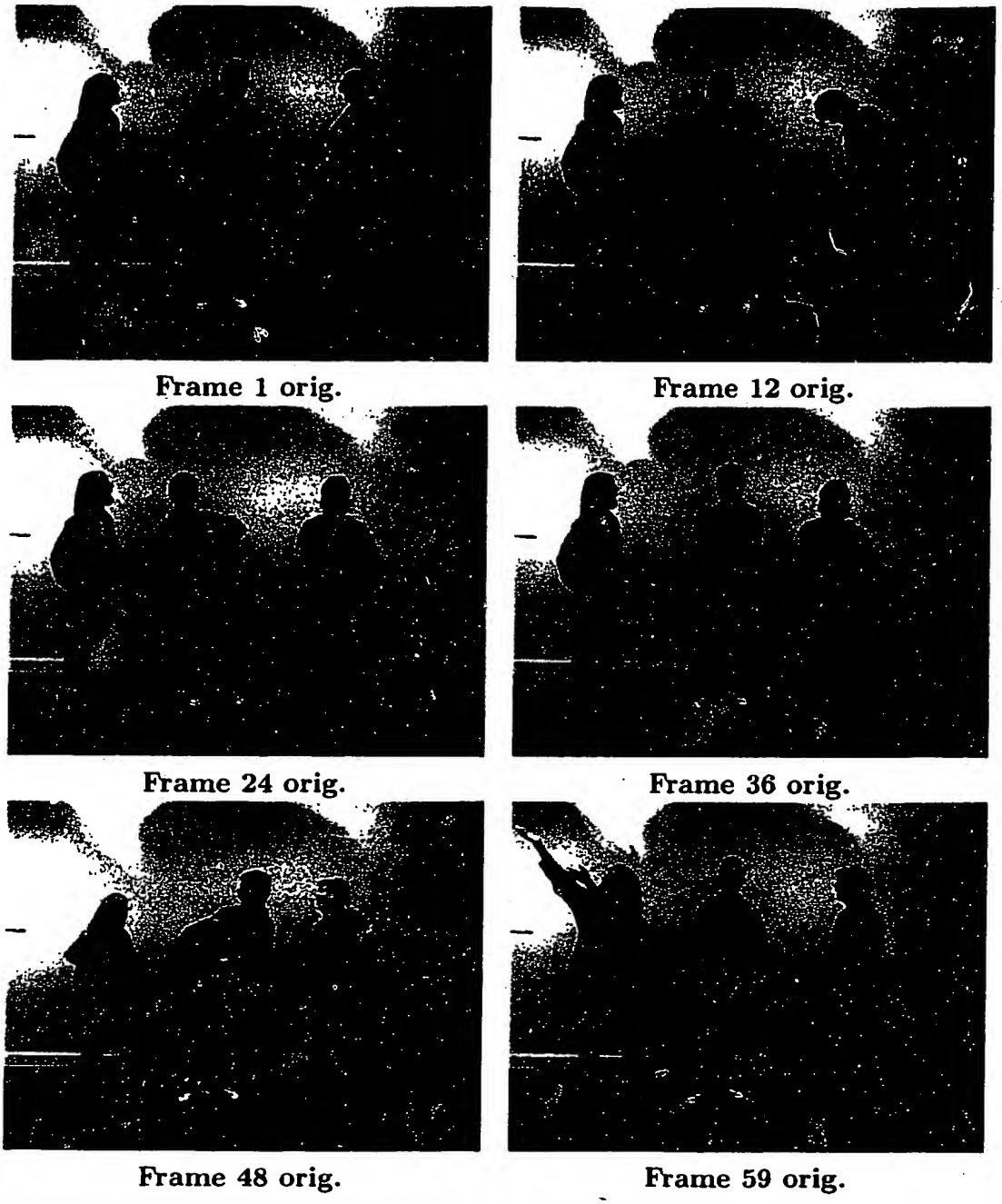
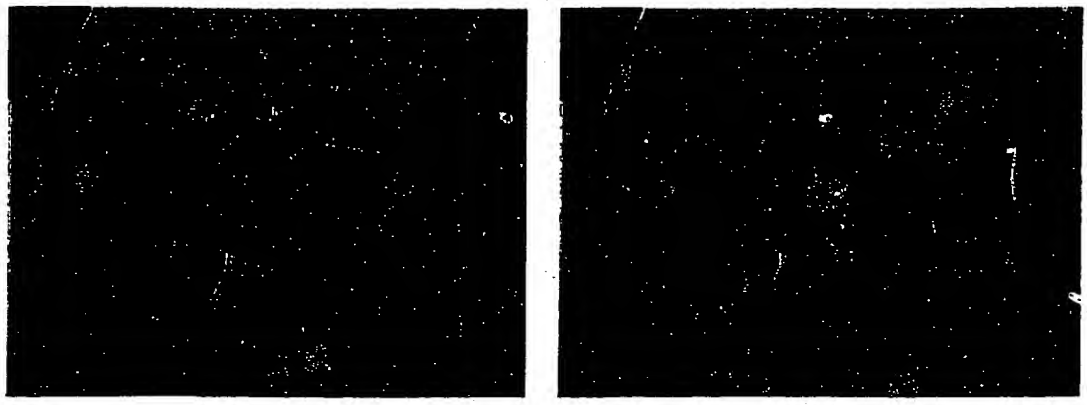
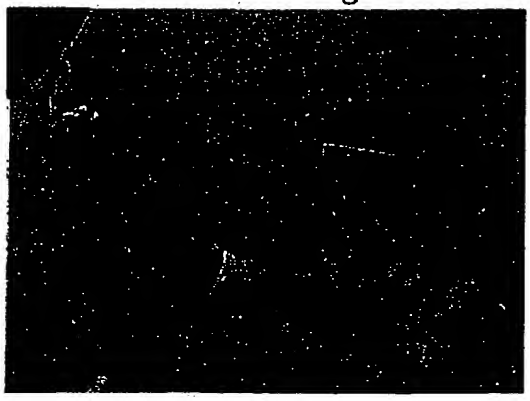


Figure 5-23: Original frames from DANCE sequence.

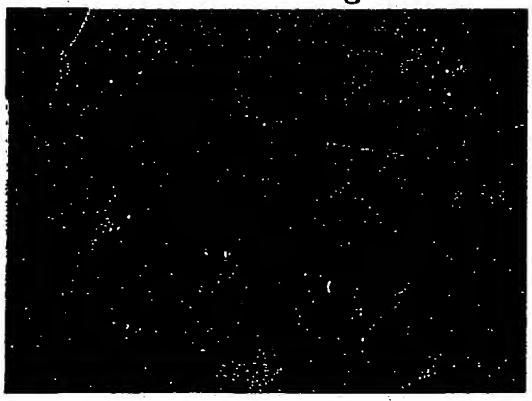


Frame 1 orig.

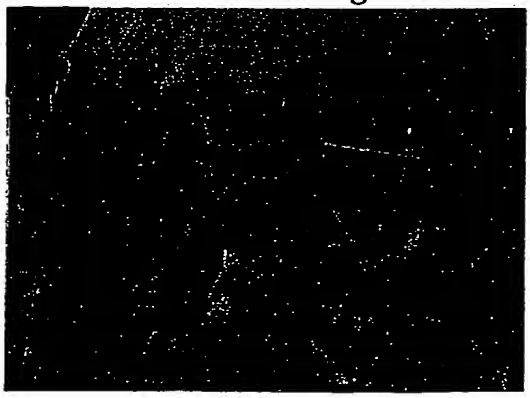
Frame 6 orig.



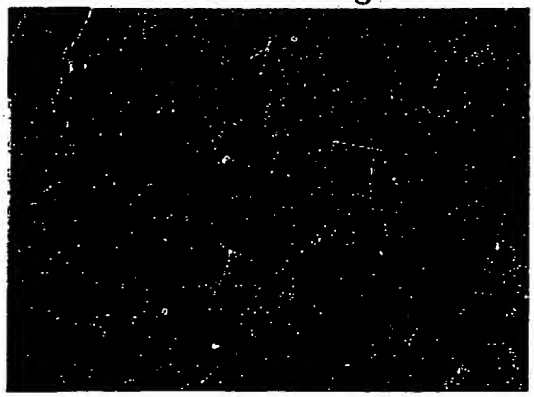
Frame 12 orig.



Frame 18 orig.

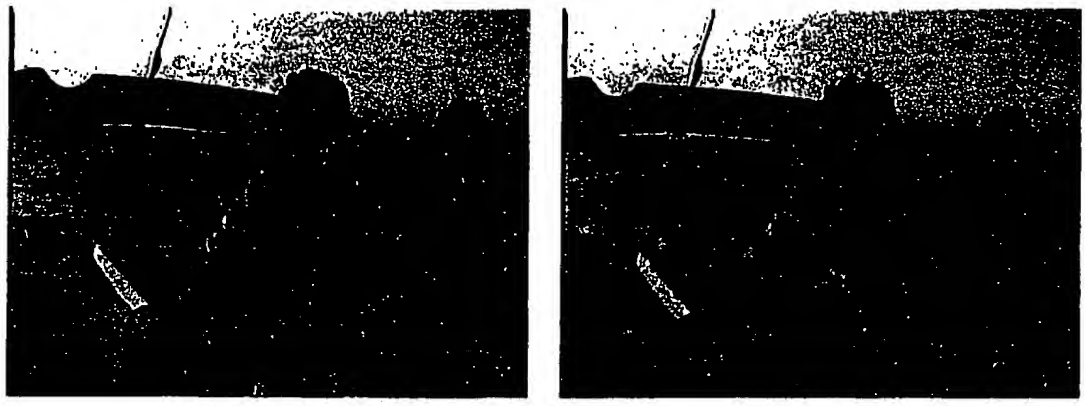


Frame 24 orig.



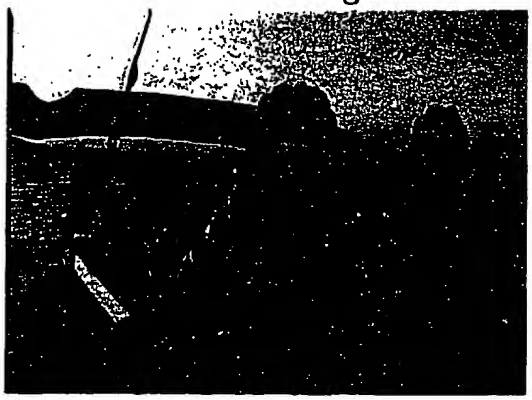
Frame 30 orig.

Figure 5-24: Original frames from JEANINE sequence.

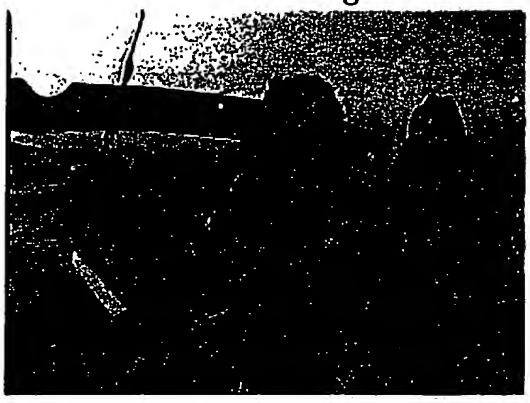


Frame 1 orig.

Frame 6 orig.



Frame 12 orig.



Frame 18 orig.

Frame 24 orig.

Frame 30 orig.

Figure 5-25: Original frames from SKIPPER sequence.

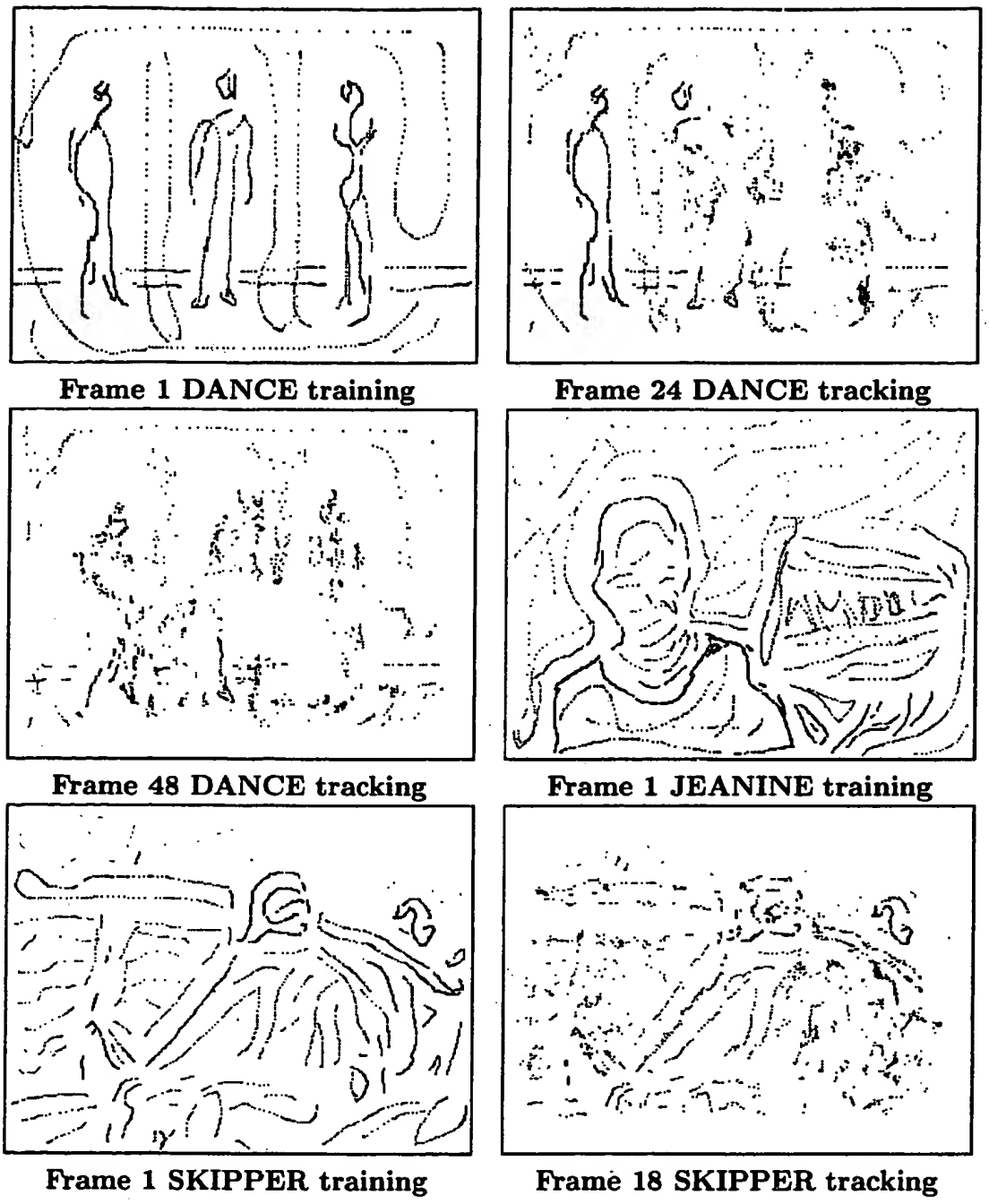


Figure 5-26: Selected frames of training and tracking of DANCE, JEANINE, and SKIPPER sequences.

each actor, also play a role. A second thing to note about the sequence (though it is not easy to note based on the still frames, one needs to see the entire sequence at a natural speed), is that the person on the right, begins facing sideways, and by the 13th frame faces forward. As a result, the tracking algorithm does not track the right side of her body (left as we are looking at it), creating big gaps in the tracking data. These gaps are the reason for the high uncertainty in Figure 5-28.

Figure 5-29 shows the segmentation results of the JEANINE sequence. And Figure 5-30 shows the segmentation results of the SKIPPER sequence. The JEANINE sequence segments out quite well, with a little bit of error in the advertisement box, between her left shoulder (on the right as we see it) and the background, and in the bottom left and right corners of the image. The SKIPPER sequence is a bit noisier, partly because of the motion in the sequence, and partly because there is quite a bit of occlusion of objects. As a result of this occlusion, even people would probably not segment out the sequence consistently between different users. For example, it is difficult to tell if the man in the background is standing in front of or behind the banister of the boat. As a result, the training data labeled the man's neck as belonging to the boat. The steering wheel of the boat also proved slightly difficult to segment, not primarily due to the motion, but rather because it is situated in the bottom right hand corner of the image, and the tracking algorithm has a boundary artifact that it can not track points in the bottom and right side of the image (as a function of the search range and block size).

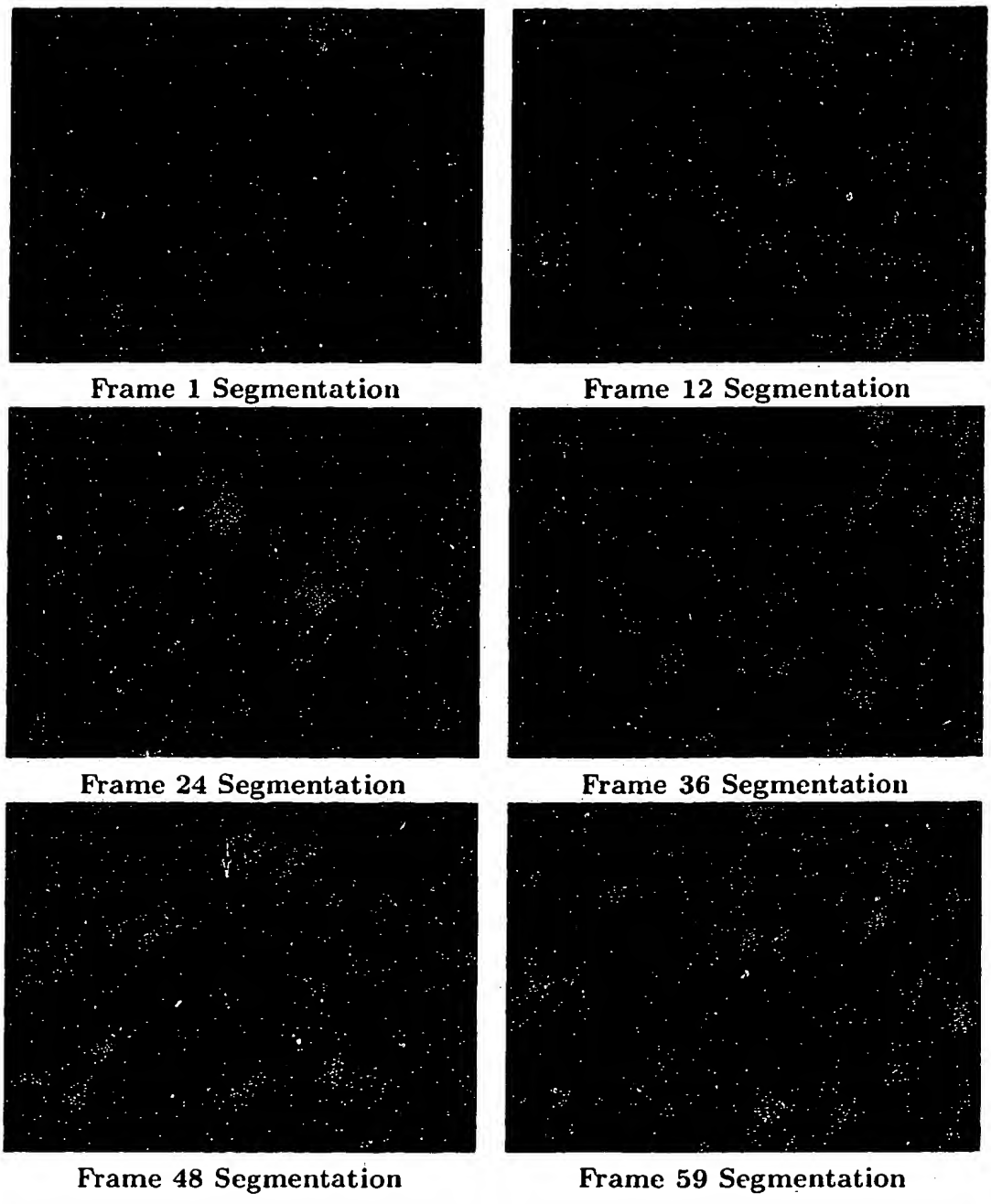


Figure 5-27: Segmentation of DANCE sequence using Method IX

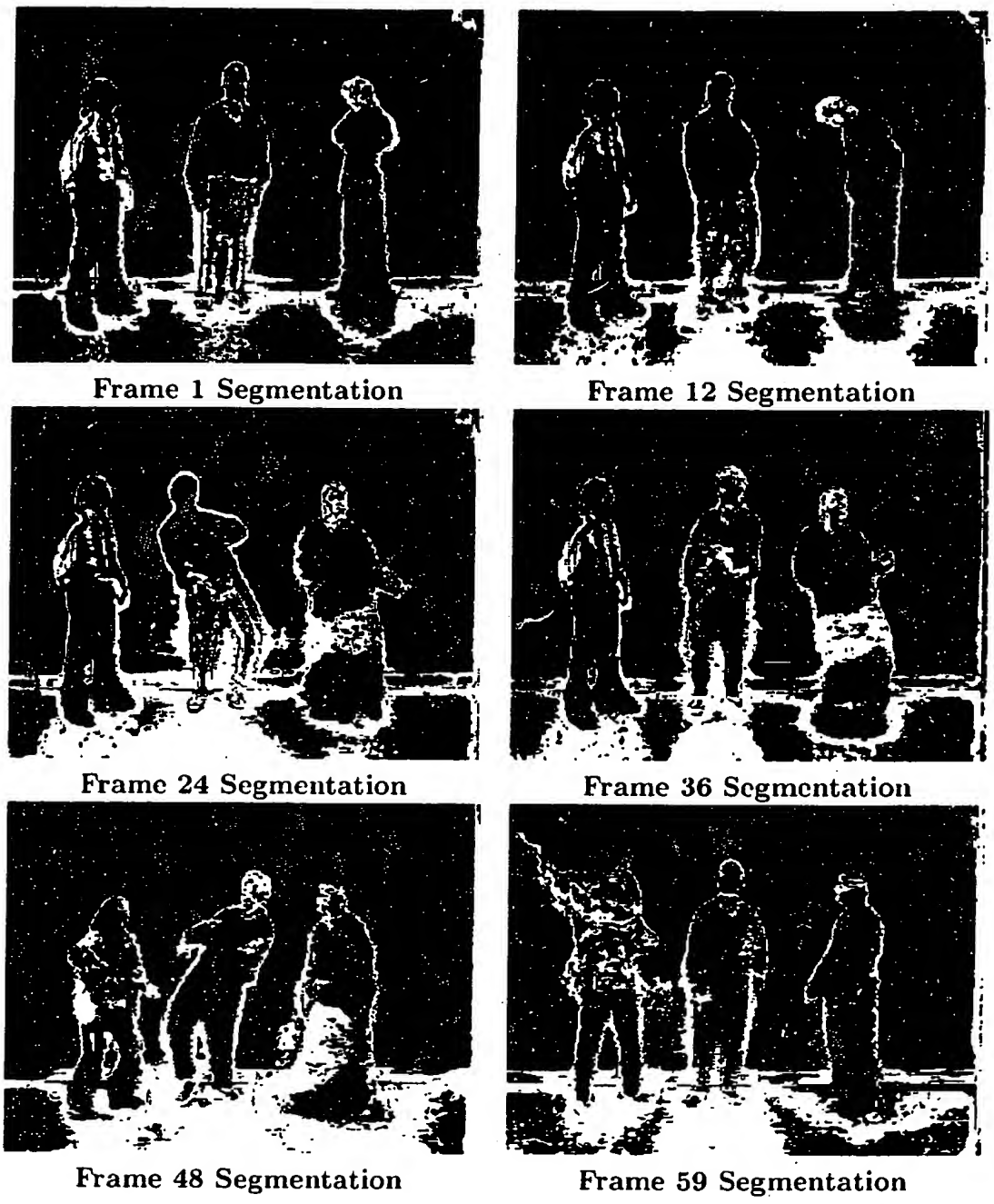


Figure 5-28: Segmentation of DANCE sequence using Method IX, certainty threshold 0.09.

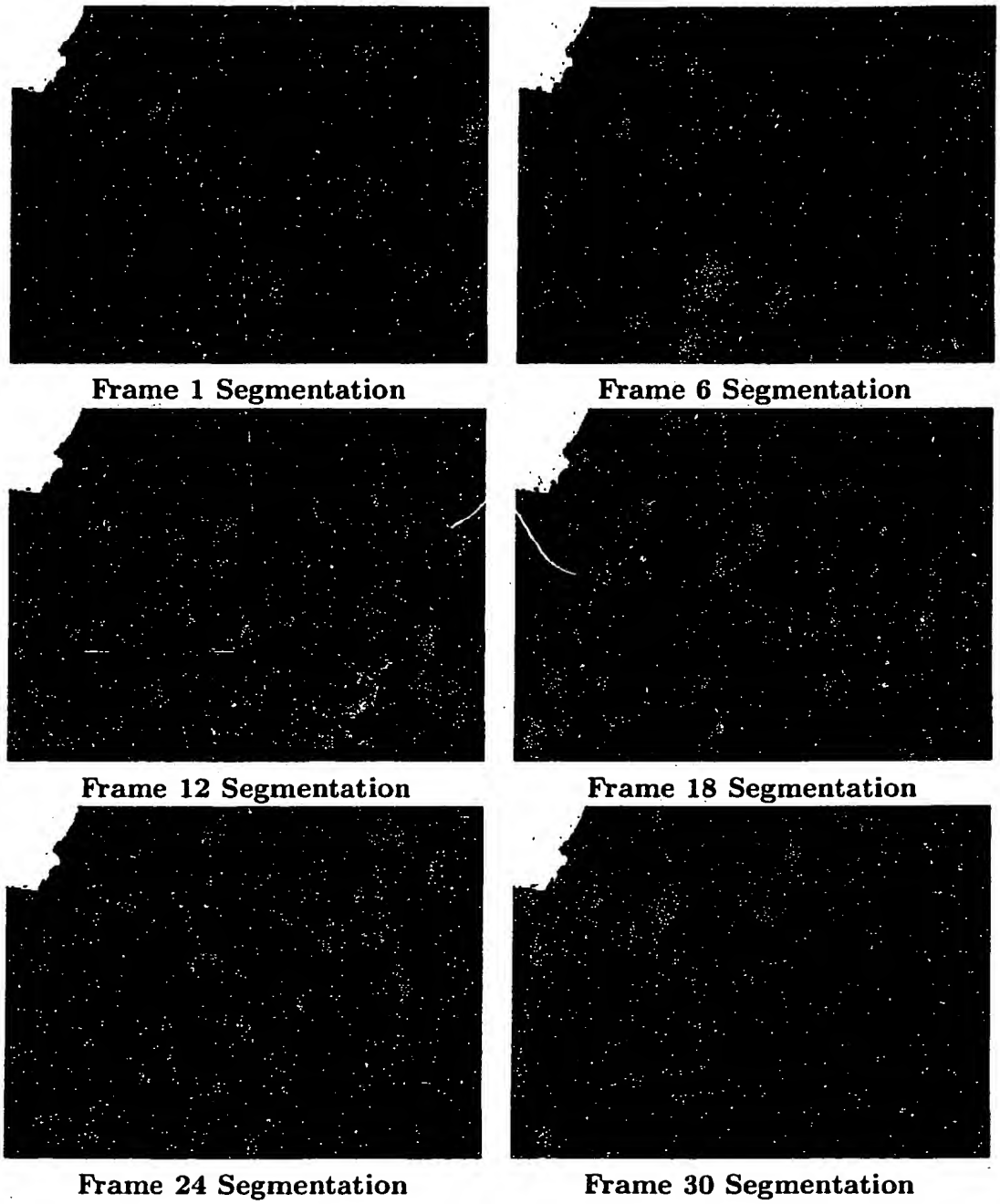
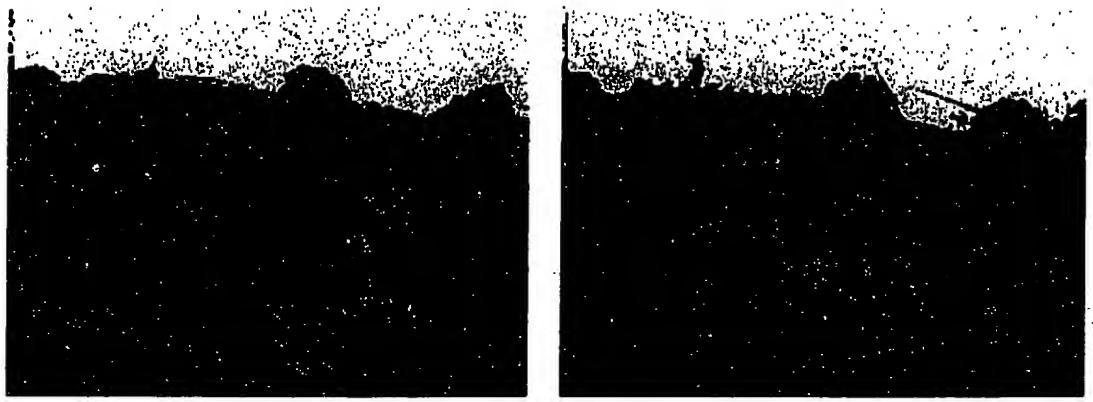
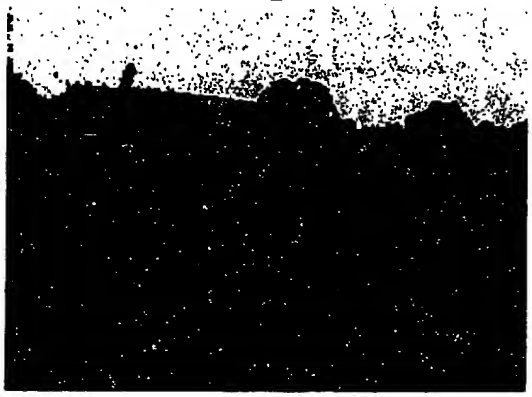


Figure 5-29: Segmentation of JEANINE sequence using Method IX.

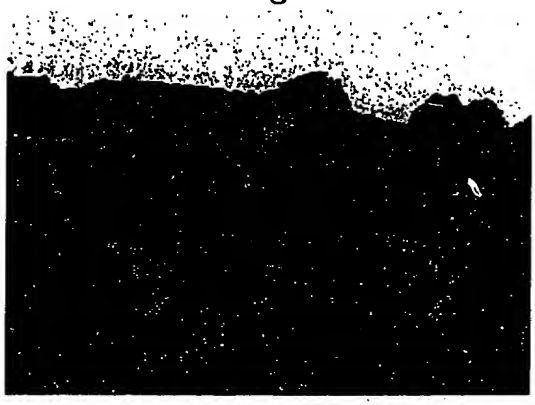


Frame 1 Segmentation

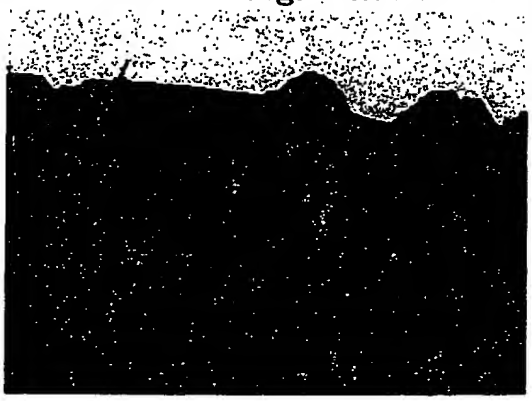
Frame 6 Segmentation



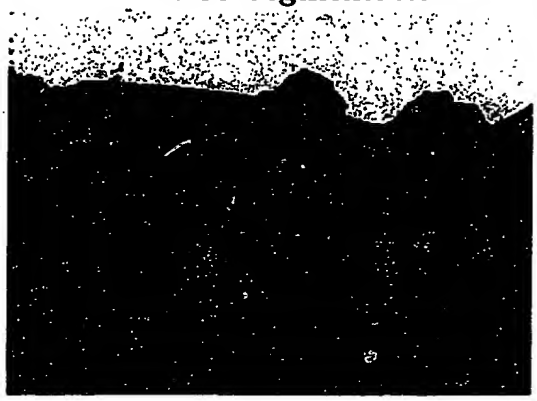
Frame 12 Segmentation



Frame 18 Segmentation



Frame 24 Segmentation



Frame 30 Segmentation

Figure 5-30: Segmentation of SKIPPER sequence using Method IX.

Chapter 6

Conclusion

This chapter will focus on a few applications that could be built with the segmentation toolbox discussed so far. Let us first reiterate some of the novel concepts in our study. Four major contributions include: the ability for our segmentation to detect user-defined regions, the study of the effectiveness of combining multiple image characteristics measured on different scales, the effectiveness of using a parametric multimodal Gaussian model to estimate the PDF, and a flexible overall modular design that allows methodology changes to easily be incorporated into the segmentation process.

First, because the regions are user-defined, different users can segment the same scene with different selection regions. Image segmentation schemes that use an unsupervised approach typically produce a fixed result per scene. Even for the schemes that provide adaptive initialization settings, they do not produce a segmentation that coincides with the user's definition of the objects in the scene. A second major contribution in our algorithm is that it utilizes conglomeration of many features simultaneously. When only one image attribute (*e.g.* color, texture or motion) is used to compute the segmentation there is a greater importance on how the feature is calculated, compared to the case when multiple features are used conjointly to compute the segmentation. By combining the features, the segmentation no longer depends exclusively on one attribute, so even simple feature calculations can produce a good segmentation. In our approach we are able to make use of a statistical framework

to combine the PDF estimation of many features to get a multidimensional estimate of each region. Moreover, because the features are calculated at every point in the image sequence, we are able to produce a densely labeled segmentation mask file with information about every pixel in the image. A third contribution is that we use a robust algorithm to estimate the PDF of each region. By using EM to solve for several multimodal Gaussian models (each with a different guess on the number of modes), and using cross validation to compare the performance of each model we can select a robust estimate of the PDF of each feature. The segmentation performance using this PDF estimation is considerably better than the segmentation performance using only a single Gaussian model. The last contribution discussed is that the segmentation algorithm is designed modularly, (see Figure 3-2) and therefore any of the stages can be improved upon separately. Specifically, if we want to use different methods to calculate the image attributes, or even add more features such as depth or 3-d shape, for example, we need only make simple changes to stage I. If we wish to change the tracking method, we would change Stage II. Stage III incorporates the PDF estimation technique, so if we want to compare how a non-parametric histogram or Parzen estimate would affect the segmentation, we would make the appropriate changes there. Finally, Stage IV takes as input the feature calculation along with the multidimensional PDF estimates from each region, and uses a hypothesis testing (Likelihood Ratio Test) to determine the segmentation. We could change this stage separately to perform a different classification operation, such as fuzzy logic or other non-Bayesian approaches.

6.1 Applications

Let us examine in detail some applications that can be built with our segmentation toolbox. Three applications on which we focus in detail include: special editing effects that enable the placement of objects from one scene into another; interactive authoring that allows video (or multimedia) to be displayed in a user-tailored order as opposed to the conventional linear frame by frame order; data compression where the

boundaries are prescribed by regions corresponding to objects, as opposed to current methods that use tile or block based regions. Other applications that could benefit from this research include problems related to machine vision such as surveillance, autonomous navigation, and medical imaging analysis.

6.1.1 Personalized Postcards / Object Editing

A great asset of having the image decomposed into regions, is the ability to edit, cut, and paste objects within an image or even from one image to another. A small project I developed to demonstrate some application of my image segmentation toolbox is called **Personalized Postcards**. The idea is that a user can choose a postcard-like image from a list of cities (target), as well as a personal image (source), and extract out objects from the source image and overlay them on the target image.

The application was built using Isis, a multi-level scripting environment that allows the programmer to manipulate complex multimedia compositions and behaviors using high-level constructs. [8] The interactive editing application allows the user some basic tools to create a composition based on one or more regions, or entire images from many images. For example, a user can scale or crop selected objects, or they can reload a new target or source image, to create composites that incorporate images or objects from multiple sources.

Figures 6-1(**a**) and 6-1(**b**) show some example original images, and their corresponding segmentation mask files. Selected objects from these images were used to create the composites shown in Figures 6-2(**a**) and 6-2(**b**). The first composite shown is called DOUBLETAKE, (Figure 6-2(**a**), top). It was created mostly for fun purposes, and as its name suggests, one needs to take a double look to notice the individual in the foreground was swapped with the statue in the center. Alternatively, the name DOUBLETAKE is because the image is shown twice. Only one original was used for this, but the ordering of the "layers" of the objects was important in creating the composite. The second composite, TUILERIES, was created from a background postcard like image (not shown) of the Tuileries Garden (Jardin de Tuileries) in Paris. Added to the scene are the GYMNAST (from Figure 5-6), as well as objects from

Figure 6-1: (a) Original and Segmentation of LOUVRE, JOE, and ERE from top to bottom

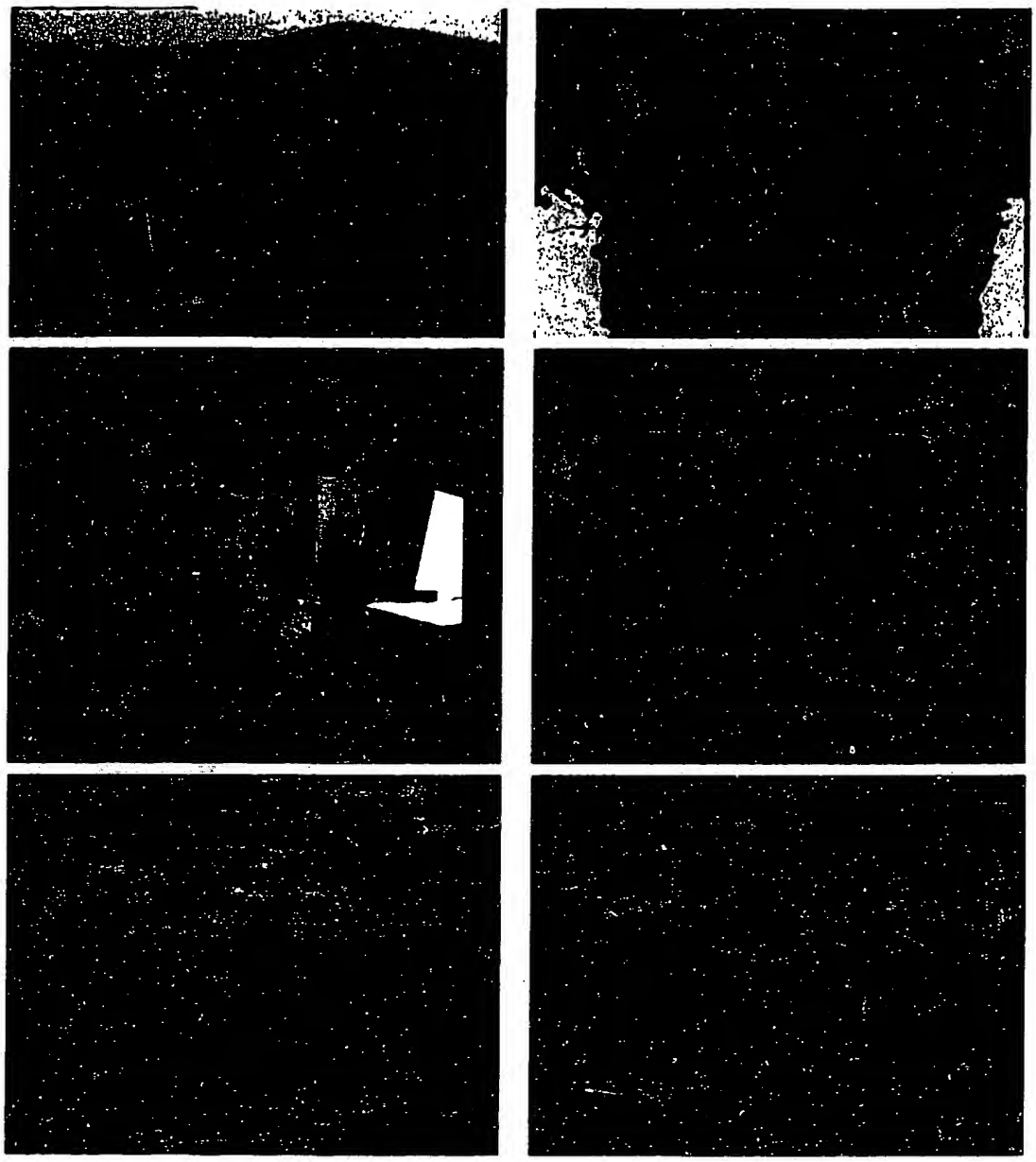


Figure 6-1: (b) Original and Segmentation of ADO, ME, and MAYER

EDDIE and MAYER images shown (2nd and 4th from top) in Figure 6-2(**a**). The sky writing is one of the features of the postcard application, the user can scale an object to a relatively small size, and use it like a paint brush. Aside from the writing, the image from MAYER is pasted 3 times in the pond. The two composite images in Figure 6-2(**b**) (ECET on the left, and EREJOE on the right) use the original ERE image (Figure 6-1(**a**) on the bottom) as its background, and they use objects taken from the ADO and JOE images, (Figure 6-1(**a**), top and Figure 6-1(**b**), middle) to edit the foreground.

The LAFAYETTE image of Figure 6-3, though not included in any of the composite images, is shown to demonstrate the quality of the segmentation. In the LAFAYETTE image specifically, there were very few training points supplied, 1290 training sample points, and an original image size of 491 rows by 703 columns, which is roughly 0.37% training points. In all of the segmentation processes, the features used were YIQ for color and local mean and variance (window size 5 by 5) for texture. Since only one frame of the original was provided, no motion information was used. Also, the KNN algorithm was used to relabel samples with low certainty.

The BALLOONS image in Figure 6-4 demonstrates how different users can provide different training points in order to obtain different segmentations. The training provided by the user would presumably extract objects of interest. In Method 1, all the balloons are treated as one object, while in Method 2 the center balloon is considered as a separate object. The resulting segmentation of both Methods perform equally well. This example truly demonstrates the benefit of allowing the user to provide training points in order to have the ability of obtaining different sets of segmentation given different sets of training. And yet, in both segmentation results, the regions still correspond well to real world objects.

6.1.2 Interactive Video Authoring

A second application known as interactive multimedia (or video) authoring, also referred to as hyper-linked video, is one that allows users to view a movie in an interactive and non-linear fashion. So instead of viewing a movie in a conventional manner,

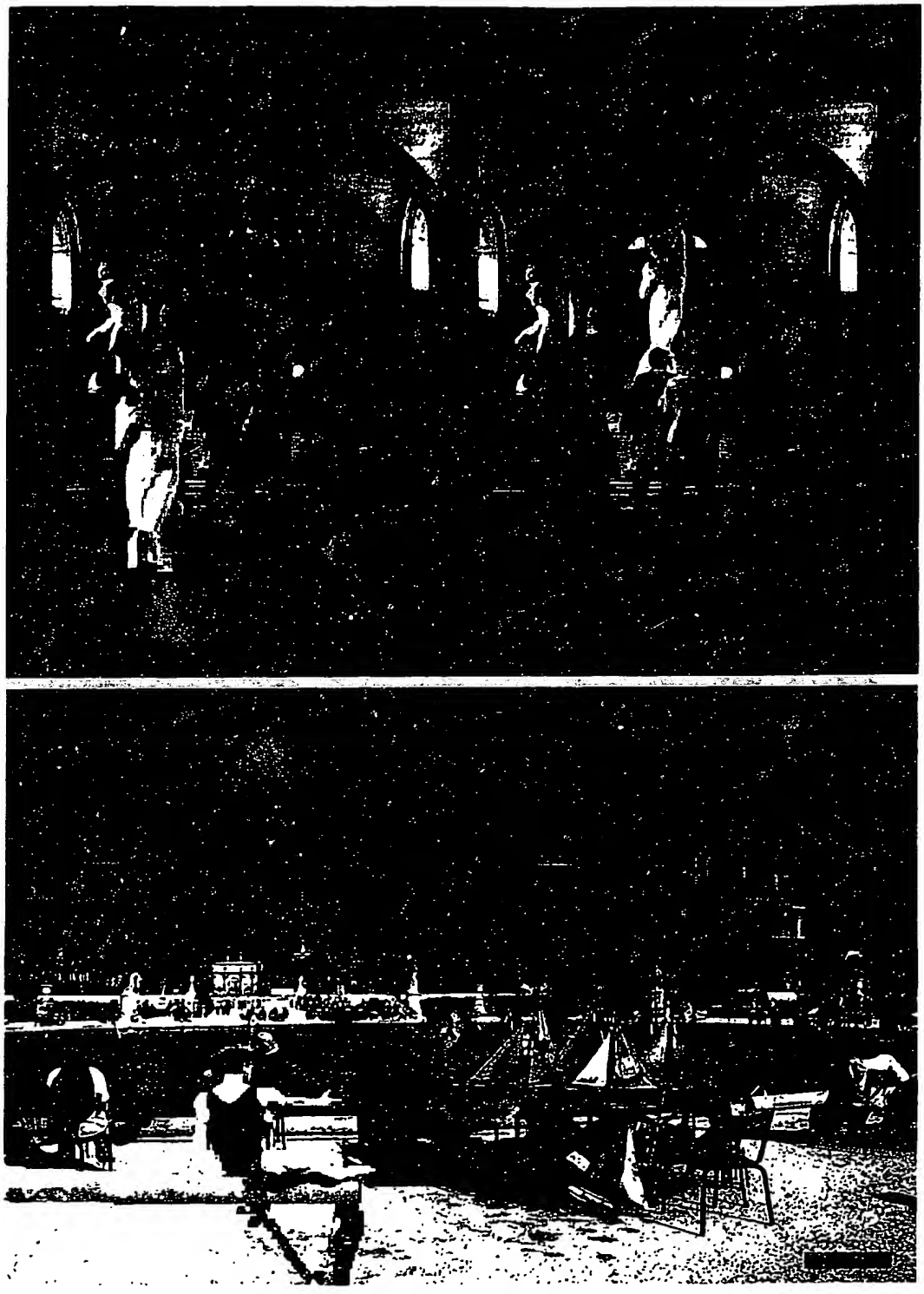


Figure 6-2: (a) Composite images DOUBLETAKE and TUILERIES

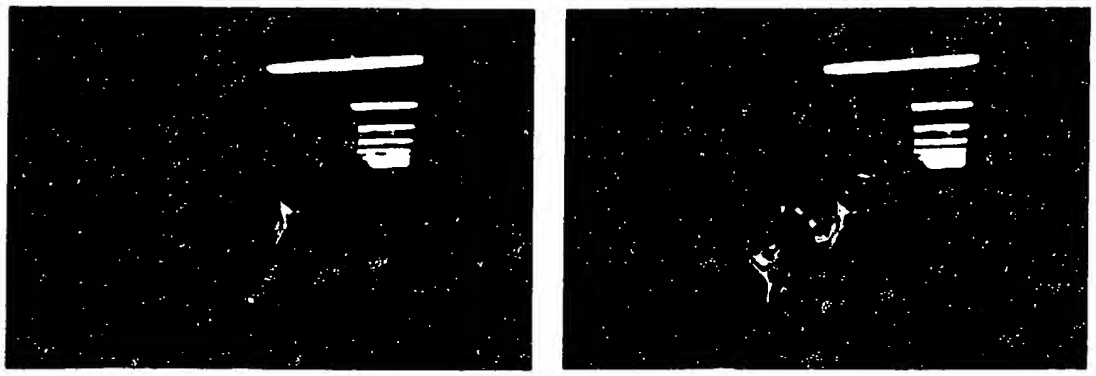


Figure 6-2: (b) More Composite Images: ECET (left) and EREJOE (right).

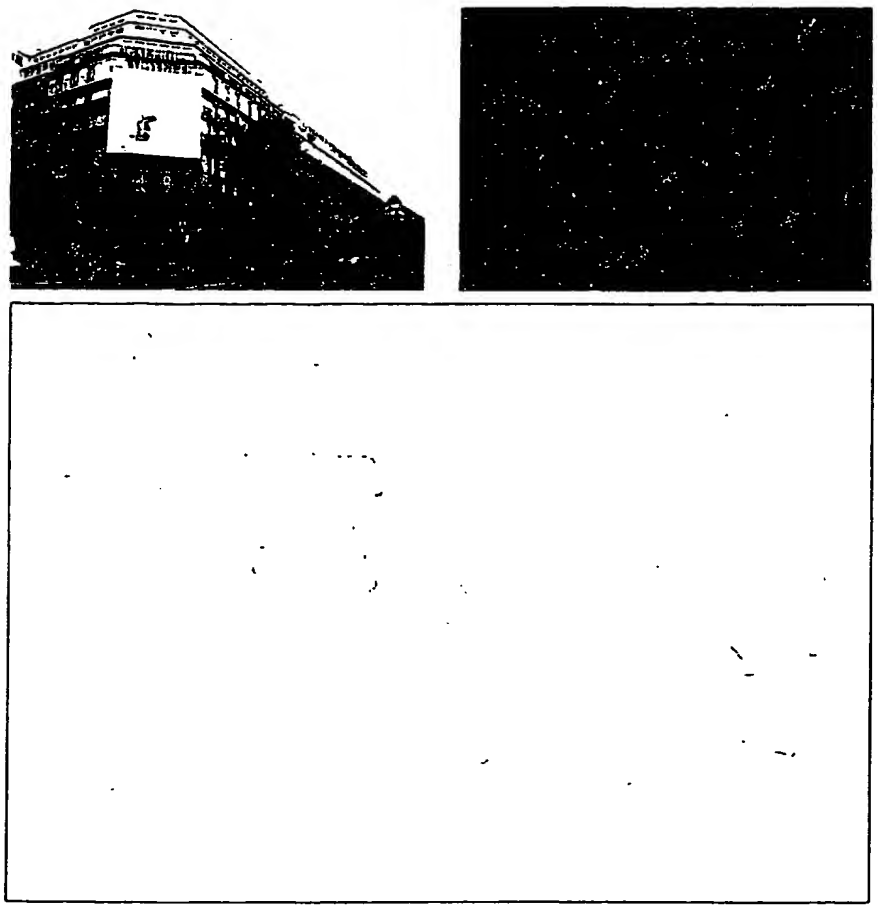


Figure 6-3: LAFAYETTE image, example of segmentation using very few training points (bottom), original and segmentation (top left and top right).

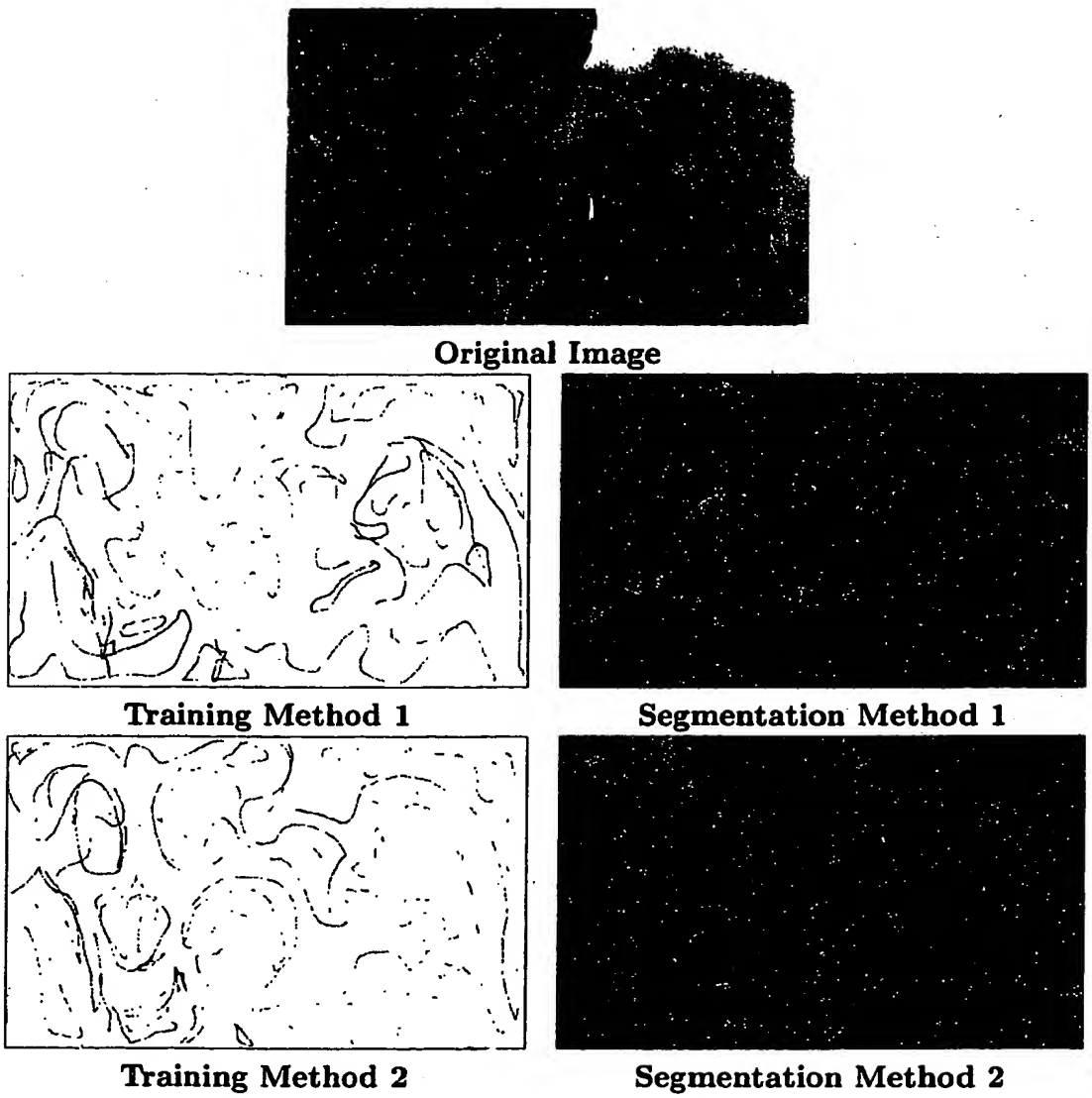


Figure 6-4: Original BALLOONS image and two methods of segmentation given different training.

from beginning to end, imagine that objects within the movie serve as "clickable" hyper-links to allow the user to "jump" or cut to another related scene containing the same or similar objects in both scenes.

The computer graphics approach to this problem is a bottoms up approach, in that objects introduced into the scene are typically synthetic or computer generated. With our semi-automatic image segmentation approach, these objects can be parts of already existing movie scenes. So multimedia authors (or directors or programmers) can present already existing multimedia in an order catered to the user.

In the personalized postcards example the generated segmentation mask files were used to edit objects. In our interactive video authoring example, the segmentation mask files can be used to help create boundaries (either as a transparent layer, or perhaps by making an object flash temporarily) in order to inform the user the location of the hyper-links.

6.1.3 Compression

The third application we will discuss here is compression. Specifically, we are considering compression applications that preserve the semantic or structural content of a video scene. Most coders historically use a block based compression technique by applying fewer bits to the higher frequency components within each block. These methods do not offer any additional semantic information provided by region based coders, and can often create block based artifacts giving synthetic boundaries between blocks.

With a region based coder, there are several advantages of having the image segmented at the encoder. For example, suppose the mask file were sent to the user/client with very low resolution (or even at full resolution) as a representation of the video sequence. Since the mask file information costs very few bits per pixel to transmit one could reduce the amount of image data transmitted by allowing the user to preview the content of a scene based on the mask file, and then allow the user to select between different image sequences. There would therefore be a great data reduction in transmitting many mask files and one "correct" image file rather than

transmitting the actual image data for all the image sequences. As an added feature of representing the image sequence in terms of objects, the user could alternatively select just an object or region of interest within an image scene based on previewing the segmented mask file. In this application too, the amount of data transmitted would be much less in transmitting the image content of one object, compared to transmitting the entire image content of frame of the image sequence.

Even without the extra pre-selection feature, the mask file can be sent with little overhead in one of the following manners: Typically it is enough to transmit the border or edge between regions, especially if the regions form closed contours. These region boundaries typically cover less than 10% of the image scene. So if they are sent directly as an integer between $1, 2, \ldots, R$, then it would take $\frac{\log(R)}{\log(2)}$ bits at the border, or an extra $0.1\frac{\log(R)}{\log(2)}$ bits per pixel per frame, to transmit the mask file information. In this method, we assume a fill function could be used at the receiver to fill in the region labels between the borders. A second approach to transmitting the mask file information assumes the video corresponding to each region will also be transmitted. If that's the case, one can send the video content of each region (one region at a time) with zero values filled in for all the samples not belonging to that current region. In addition, a bounding box or crop could be used to include only the necessary portion of the frame that bounds the region of interest. In theory, this would add very little energy to each transmitted region, except perhaps a little bit of high frequency energy at the border of the region. Therefore, very few additional bits per pixel would be required to encode these additional zeros. And thus each region's mask could be decoded at the receiver by finding the non zero values, one region at a time.

6.2 Future Research / Improvements

With an experimental topic such is this, there is always more research to be done. Here are a few specific examples of modifications that can be made. First, at the feature calculation stage, it would be great if some kind of preliminary detection could be made on the input image and/or the training data, to automatically decide

which combination of image attributes and the method of calculating them would result in the best image segmentation. One quick way to do this is to perform a first order PDF estimation, such as a histogram, and define some kind of distance metric between the PDF of each region. The combination of features with the maximum distance between PDF estimates would be a good candidate for the choice of features in the segmentation.

At the tracking and training stage, it would be extremely useful to have a bi-directional tracking algorithm, or alternatively a feedback mechanism to allow the user to correct the tracking points. Currently, a multi-dimensional PDF estimate is required for each user-defined region. The base frame estimate is created from the user supplied training data, and the estimate can be updated based on the tracking algorithm. For long term sequences, say on the order of a thousand frames instead of a hundred, the user should be able to provide training data for the any of the regions at any point in the one long sequence, especially since there could be some regions that are only visible in part of the sequence. The difficulty is in deciding how often or uniformly the user provides training data points for each regions. Allowing for training at multiple frames would also certainly affect the tracking procedure. Specifically, if training for a region is provided at both frames 1 and 100, then a bidirectional tracking algorithm should use the training information from both frames to estimate the location of these training samples at intermediate frames. Additionally, perhaps there could be some kind of feedback in the training/tracking process. Specifically, after tracking the regions the user can scan through the tracked points over time, perhaps as a transparent layer overlaid on the original sequence, and edit the training data where necessary.

Another improvement for the system could be in reducing the computational time. The biggest bottle neck of the system is by far at the PDF estimation stage. This is because the EM algorithm in itself is an iterative algorithm. Add to that the fact that the EM algorithm is used to compute the best multimodal fit for each of $M = 2, 3. \ldots, 8$ modes in order to compare them all and find the best model. A more computationally effective model, at the cost of quality, would be to use a unimodal

Gaussian estimate or a histogram method. More research is necessary to determine the quality of a Parzen estimate.

Yet another aspect of the segmentation scheme that could use more research is the decision criterion used for labeling, given the PDF estimations of each region. Currently we apply the "standard" hypothesis testing method which uses a likelihood ratio test to decide which region a pixel belongs to, give the PDF estimates of each region. However, this method does not take full advantage of the multi-dimensional information available. Specifically in a small percentage of data points, a sample point may have a very high likelihood of belonging to a region (relative to the other regions) given the PDF estimate of only one attribute (color, for example), but that same sample may have an almost impossible likelihood to fit that same region given another attribute (position, for example). A rule based method can allow very unlikely values to, in effect, veto that region from being selected as the most likely region. Though in our current method, the likelihood values of each attribute are combined (multiplied), and the region with the maximum combined likelihood is selected. So in some instances, the most likely region is selected despite the fact that it has an almost impossible likelihood along one dimension of the feature space.

In the long term, it is fairly certain that any robust image analysis system of natural video scenes must decompose the scene into regions or objects in order to extract semantic information. Our research can serve as a preliminary building block along that path. In particular, these machine vision systems should not depend exclusively on statistical methods. Rather, a higher order knowledge data base could be used in conjunction with our lower order building blocks. This higher order data base can be built upon a large variety of objects segmented from a large data base of image sequences. The data base could perhaps store inferential information, or properties about these regions, and in turn provide useful information back to segmentation process. Or it can provide key information for performing image-based searches or indexing of data bases. With more experimentation and research, we will hopefully gain a deeper understanding of autonomous image understanding.

Appendix A

RGB to CIE-XYZ conversion

We want to solve for the coefficients of the matrix M, m_{ij}, $1 \leq (i,j) \leq 3$, that converts a set of RGB values to a set of XYZ values that represent the 1931 CIE-XYZ primary color system. The Y value represents the luminance information, and the X and Z values represent the color information. The XYZ values are sometimes referred to as **tristimulus values.**

$$\overbrace{\begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix}}^{M} \begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{A.1}$$

Let us define

$$x = \frac{X}{T} \tag{A.2}$$

$$y = \frac{Y}{T} \tag{A.3}$$

$$z = \frac{Z}{T} \tag{A.4}$$

where, $T = X+Y+Z$, is a normalization factor, and the (x, y, z) values are referred to **chromaticity coordinates** and they represent the amounts of the XYZ primaries.

Because we have normalized the (x, y, z) values, they are constrained to $x+y+z = 1$. We use this normalization constraint along with Equations A.2 and A.4 to show that:

$$X = \frac{x}{y}Y \tag{A.5}$$

$$Z = \frac{z}{y}Y = \frac{1-x-y}{y}Y \tag{A.6}$$

Thus, we only need the (x, y, Y) triplet to uniquely define (X, Y, Z). (A **chromaticity diagram** plots the luminance Y, as a function of the chromaticity coordinates (x, y).)

For a given display, we are always told the chromaticity coordinates of white (balance), and red, green, and blue. In RGB space, we also know the (R, G, B) values of these four colors. Namely $(R, G, B) = (1, 0, 0)$, $(R, G, B) = (0, 1, 0)$, $(R, G, B) = (0, 0, 1)$, and $(R, G, B) = (1, 1, 1)$ for each of red, green, blue, and white respectively. And we know the luminance value, $Y = 1$ for white. Let us define the (x, y, Y) triplet values for each of red, green, blue, and white as (x_r, y_r, Y_r), (x_g, y_g, Y_g), (x_b, y_b, Y_b), and (x_w, y_w, Y_w),

By using the above RGB values along with Equations A.5 and A.6, we can derive the $m_{i,j}$ values from the matrix M in Equation A.1 as a function of the (x, y, Y) triplet values for each of red, green, blue, as

$$M = \begin{bmatrix} \frac{x_r}{y_r}Y_r & \frac{x_g}{y_g}Y_g & \frac{x_b}{y_b}Y_b \\ Y_r & Y_g & Y_b \\ \frac{1-x_r-y_r}{y_r}Y_r & \frac{1-x_g-y_g}{y_g}Y_g & \frac{1-x_b-y_b}{y_b}Y_b \end{bmatrix} \tag{A.7}$$

Recall that for a given display, we are only given the (x, y) values, but not the Y values for red, green, blue and white. To solve the (Y_r, Y_g, Y_b) values, we evaluate Equation A.1 at reference white $(R, G, B, Y) = (1, 1, 1, 1)$, using our matrix M, from

Equation A.7, we write

$$\overbrace{\underbrace{\begin{bmatrix} \alpha_r & \alpha_g & \alpha_b \\ 1 & 1 & 1 \\ \gamma_r & \gamma_g & \gamma_b \end{bmatrix}}_{P} \begin{bmatrix} Y_r \\ Y_g \\ Y_b \end{bmatrix}}^{M} = \begin{bmatrix} \alpha_w \\ 1 \\ \gamma_w \end{bmatrix} \tag{A.8}$$

where we define $\alpha = \frac{x}{y}$, $\gamma = \frac{1-x-y}{y}$, and P is the matrix as shown in Equation A.8.

In our XYZ-RGB conversion, we use the following reference values: The white balance comes from a standard known as D6500. The red, green, and blue values are specified by the monitor (or some that standard to which the monitor prescribes). The chromaticity values we use for white, red, green, and blue, respectively are:

$$\begin{aligned} (x_w, y_w) &= (0.3130, 0.3290) \\ (x_r, y_r) &= (0.6280, 0.3460) \\ (x_g, y_g) &= (0.2680, 0.5880) \\ (x_b, y_b) &= (0.1500, 0.0700) \end{aligned}$$

So we can now compute the values of the matrix P as well as the α_w and γ_w coefficients. Using our relation in Equation A.8 we can solve directly for the (Y_r, Y_g, Y_b) vector simply by multiplying both sides by the P^{-1} matrix, which yields $(Y_r, Y_g, Y_b) = (0.2635, 0.6550, 0.0815)$. Thus we now know the (x, y, Y) for each of red, green, and blue. Plugging these values into Equation A.7, we find that

$$M = \begin{bmatrix} 0.4782 & 0.2986 & 0.1746 \\ 0.2635 & 0.6550 & 0.0815 \\ 0.0198 & 0.1604 & 0.9079 \end{bmatrix} \tag{A.9}$$

which is the same as our value for M in Equation 3.25. The XYZ primary values can easily be converted to other color spaces, such as LAB in our experiments.

Appendix B

User's Manual

This appendix is intended to assist anyone who might need to execute the segmentation programs installed on DEC Alpha workstation in the MIT Media Laboratory. You may find it helpful to refer to

B.1 Block Diagram

The reader may find it helpful to refer to Figure 4-1, which contains a block diagram of the segmentation process. Also, Chapter 4 provides a more general description of what the programs are designed to do. Both this appendix and Chapter 4 have the same section names to make it easier to follow both simultaneously.

B.2 Feature Calculation

The first stage of the block diagram in Figure 4-1 deals with calculating the feature information of motion, texture, color and position.

B.2.1 Texture

The three methods we've implemented to calculate the texture feature are the local statisitics method, the multiresolution statistics method, and the simultaneous autoregressive method.

Local Statistics

The program to compute local texture statistics is located in

`/mas/garden/grad/echalom/src/texture/local_stats`

and it is run with the following parameters

`local_stats <fin> <statout> <y_win_size> <x_win_size> [-rgb_y]`

The `<fin>` parameter is the input file name, and the `<statout>` parameter is the output file name. The `<y_win_size>` and `<x_win_size>` parameters specify the local horizontal and vertical window size. Experiments were run with varying window sizes which were usually between 3 and 15 pixels. The `[-rgb_y]` option signals the program that the input file is an rgb file, while the output file shout be the local statistics of the luminance value only. Otherwise, if the `[-rgb_y]` option is not used (which is also the default), then the output file will contain the local mean and variance for each channel of the input file. For example, suppose the input file is an RGB file with 3 channels, then the output file will contain 6 channels, corresponding to the local mean and standard deviation for each of the red, green and blue channels. However, if the input file contains only the luminance information, then the output file will be 2 channels, corresponding to the mean and standard deviation of the luminance channel. The number of data sets (frames), as well as number of rows and columns remains the unchanged between the input and output files. The output file values for both the local mean and local standard deviation channels are in the $0 - 255$ range.

Multiresolution Statistics

To compute the multi-resolution texture parameters, a shell script was written which runs the `local_stats` program with 3 different window sizes (typically 3, 9, and 15 pixels). Each execution of the local statistic program creates a 2-channel output file. However, for the multiresolution statistics texture parameters, only 4 of the 6 channels were used. Specifically all 3 channels corresponding to the local mean (each from a different window size) are included in the multiresolution texture parameters.

Only 1 channel corresponds to the local standard deviation, specifically that of the middle window size. Therefore the output texture file contains a total of 4 channels. A sample shell script file that calculates the multiresolution texture parameters is located in

`/mas/garden/urop/spazzoid/ImageSeg/mktexture_multi`.

The number of data sets, rows, and columns all remain the same between the input and output files. The output file values for all channels are in the $0 - 255$ range.

Simultaneous Auto Regressive Model

The program that computes the SAR texture parameters is located in

`/mas/garden/grad/echalom/src/texture/sar/sar`.

and it is run with the following parameters

`sar <image> <sar_texture_file> [-xwin <size>] [-ywin <size>] [-verbose]`

The `<image>` parameter corresponds to the input image file name, and is a one channel luminance signal. The `<sar_texture_file>` is the output data file which will contain five channels, one channel for each of the SAR coefficients as described in section 3.2.2. The first four channels correspond to the coefficients a_1 through a_4, respectively, and the last channel corresponds to the w, or error coefficient. The `[-xwin <size>]` and `[-ywin <size>]` parameters are optional, and the default value is 10 for each. The size, if specified must be an integer greater than 1 in order to avoid singular matrix in the least squares fitting, when calculating the five SAR coefficients per window block. The full window block size is one plus twice the specified parameter value, for each of the horizontal and vertical directions. The output file values for the first four channels are between -1 and 1, and the last channel is between -255 and 255. The number of data sets, rows, and columns all remain the same between the input and output files.

B.2.2 Color

The three methods we've implemented to calculate the color information are RGB, YIQ, and LAB.

RGB

The original image sequence is typically in an RGB format, so no transformation program is required for this feature. The file is a 3 channel data set, the first channel contains the red pixel values, the second corresponds to green, and the third channel to blue. The values for all three channels are in the $0 - 255$ range.

YIQ

To convert an RGB file to YIQ, there is a program called `rgbyiq` which has been installed on the garden computer system, located at

`/system/local/bin/$SYS/rgbyiq`

and is run with the following parameters:

```
rgbyiq <infile> <outfile> [-y] [-i] [-q] [-v] [-t abcdDfFuUzZ]
```

The `<infile>` is a 3-channel RGB file, and `<outfile>` is the name of the output file, where the number of channels depends on which optional parameters are used. The `[-y] [-i] [-q]` options specify which of the channel(s) to calculate. The default setting, *i.e.* specifying none results in all three channels to be calculated. The `[-v]` option means a verbose printout will be displayed as the program is executed. And the `[-t]` option with the appropriate choice specifies the file type to save the datfile. In our experiments, the files were specified as float so the `[-t f]` option was used when running, and all three of the Y,I and Q channels were calculated and used as features. The data file contains the Y, I, and Q channels in that order. The range of values for the Y channel is $0 - 255$, and the range for the I and Q channels are each from -128 to 127. All of the input and output dimensions remain the same.

LAB

To convert an RGB file to LAB, the program used is

```
/mas/garden/urop/spazzoid/ImageSeg/rgbtolab <dinrgb> <doutlab>
```

The <dinrgb> parameter specifies the input RGB datfile, and <doutlab> specifies the output LAB datfile. The output data type is stored as floating points, and the range of values of the LAB file are from 0 to 100 for the first channel, (L), and from -128 to 127 for the last two channels (A and B). All of the input and output dimensions remain the same.

B.2.3 Motion

The two mostion estimation techniques we've implemented are the Lucas and Kanade optical flow method, and the Horn and Schunck optical flow method.

Lucas and Kanade

The program that calculates the optical flow fields based on the Lucas and Kanade method can be found at

```
/mas/garden/staff/dufaux/bin/alpha_osf3/lucas_kanade
```

and it is run in the following format:

```
lucas_kanade <infile1> <infile2> <speedfile> [-d <deriv> (2 or 4)]
        [-p prefilter] [-s sigma] [-t abcdDfFuUzZ] [-v]
```

The input parameters specified by <infile1> and <infile2> must each be a single frame datfile. The <speedfile> parameter is the name of the output file, which contains the velocity estimation information. The optional parameter [-d <deriv>], where <deriv> is either 2 or 4, specifies how many spatial pixels should be used when computing the horizontal and vertical derivatives. If not specified, the default <deriv> value is 2. The [-p] option applies a Gaussian prefilter to the image. The same filter is applied separately in both the horizontal and vertical directions. If

the [-p] option is specified, the [-s sigma] option can also be used to specify the standard deviation of the Gaussian prefilter. Unless the prefilter option is specified, the default is no prefiltering, and if it is specified, the default sigma value is 0.5. The [-t] option specifies the data type of the output file, the default is float. The [-v] option gives a verbose printout of the settings that have been set. The output file has the same number of rows, columns, and channels as the input file, but the number of data sets for the output file is 2. The first frame corresponds to the horizontal motion field estimate, and the second frame to the vertical motion field.

Since the program only finds one motion field per pair of frames, a shell script is used to evaluate the motion field for an entire input image sequence. The output data file sequence has the following dimensions: The number of frames in the motion sequence is one frame less than the length of the original sequence. The motion sequence consists of two channels, the first contains the horizontal motion field, and the second contains the vertical motion field. The number of rows and columns remains the same as the original image. In our experiments, the program was run with the default settings, namely no prefilter, and a <deriv> value of 2. An example of a shell script that generates a Lucas and Kanade optical flow motion field sequence can be found at

/mas/garden/urop/spazzoid/ImageSeg/mkopflowlk

Horn and Schunck

The Horn and Schunck optical flow algorithm can be found in

/mas/garden/staff/dufaux/bin/alpha_osf3/horn_schunck

and is run with the following parameters:

horn_schunck <infile1> <infile2> <speedfile> [-i <itermax>]
[-a <alphasqr>] [-d <deriv> (2 or 4)] [-p prefilter] [-s sigma]
[-t abcdDfFuUzZ] [-v]

There are only two additional optional parameters in this algorithm compared to the Lucas and Kanade motion estimation. The [-a <alphasqr>] parameter is a coefficient that controls how much weight is applied to the global smoothness constraint, and has a default value of 100. The [-i <itermax>] parameter specifies how many iterations are to be used when calculating the motion estimation parameters, and has a default value of 100. All the other parameters in the Lucas and Kanade program are in the Horn and Schunck program, and are use the same way. The output data file format and dimensions are the same for both programs as well, as are the minimum and maximum legal values. A shell script was used to generate a motion estimation sequence for the Horn and Schunck method, analogous to the Lucas and Kanade method. An example shell script can be found at

`/mas/garden/urop/spazzoid/ImageSeg/mkopflowhs`.

B.3 Training and Tracking

There is one program for each the training and tracking procedures.

B.3.1 Training

The program used to acquire the training points is located in

`/mas/garden/nick/echalom/XmDat24_with_lasso/train_image`

and is run with the following parameters

```
train_image <datfile> [-oi <output index file>]
[-om <output mask file>][-bits <display bits>]
        [-quality <of dithering>] [-install]
```

The program uses X-Motif toolkits, and receives keyboard and mouse commands to execute the training acquisition. Specifically, a window pops up to display the input image datfile, and the user selects training points by highlighting the points of interest by dragging the mouse with the left mouse button depressed. A new region is specified

by clicking on the right button. Thus, sample points from the same region may be discontinuous. The user can drag the mouse (with the left button depressed) over a series of points, release the left mouse button, move the mouse over to a different area of the picture (but which still belongs to the same region) and continue to select more points by dragging the mouse (while depressing the left button) over the points of interest. A new region is introduces only by clicking on the right button. When all the points are selected, the user presses the 'q' on the keyboard to quit. Column, row and region information of all the selected points are stored into an index datfile, and or a mask datfile, (explained below) or are printed to standard output.

The `<datfile>` parameter should be a single frame RGB datfile, If neither the `[-oi]` or `[-om]` option is specified, the selected points are not saved, but rather are printed out to standard output. This option was not used when running experiments, but rather just for testing purposes.

The `[-oi <output index file>]` is an optional parameter that specifies the name of the output index datfile. The index datfile contains the column, row, and region information of all the user-selected points. It has one data set, one channel, 3 columns and P rows, where P is equal to the total number of training points selected by the user. The first column of the index datfile corresponds to the column position of the training points (that was selected by the user). The second column of the index datfile corresponds to the row position of the training points. And the last column corresponds to the region number (with values $\{0,1,\ldots,R-1\}$) of the training points. The values of the 3rd column of the index datfile data range from 0 to $R-1$, where R is the total number of regions specified implicitly by the user. Thus all the selected training points can be recovered from the index datfile.

The `[-om <output mask file>]` is an optional parameter that specifies the name of the output mask datfile. The mask datfile and the index datfile are similar in that they both contain the row, column, and region information of the training data. But they differ in their format of storing the information. The index datfile is a list containing the column, row, and region information from each training point. The mask datfile, however, is a graphical representation of this information, with a

different color for each of the selected regions. Black represents unlabeled points, or points that were not selected in the training acquisition. (As an aside, black is used to represent unlabeled points on a display monitor, but white is used if the image is printed to paper.) The mask datfile therefore, has the same number of rows and columns as the input datfile, and is one data set and one channel. The range of values of the mask datfile is 0 to R, where R is the number of regions, and 0 corresponds to an unlabeled sample. Thus the values in the mask datfile are offset by 1 when compared to the corresponding data in the third column of the index datfile. In order for the mask datfile to be displayed in color, a lut file, as described in Table 4.1 is added (either in a shell script or at the command line) using standard datfile utility library functions.

The last 3 parameters, `[-bits]`, `[-quality]`, and `[-install]` are optional parameters that specify the display settings. The appropriate values for these parameters depends on the display properties of the system on which the program is run. However, almost always, it is enough to omit specifying these parameters.

The `[-bits]` value must be between 1 and 8 if the input datfile is 1 channel, and between 3 and 8 if the datfile is 3 channels. Or it may be set to 24 in the case of a true 24 bit display is in use, and the datfile is 3 channels. The default value is set to 8. The `[-quality]` value maybe set to either 0, 1, or 2, and specifies a different type of dithering method: decimation, ordered dithering, and Floyd Steinberg Error Diffusion, respectively. The default value is 2. The `[-install]` option specifies that a new color map should be installed to display the datfile and generally displays the datfile better, but takes slightly more time to load. Otherwise, a best match from the currently loaded colors in the X-display will be used. The default is no color map installation.

Again, note that the output index datfile and the output mask datfile, both contain the same information. They differ in that the index datfile is useful as a list, to be passed to other programs, while the mask datfile is useful to be displayed (with the use of a lut file). Before we continue to describe the training program, let us take a break to discuss two utility programs that convert a maskdatfile to an index datfile and

vice versa. The utilities are useful in order to pass the resulting file from the tracking program as an input file to the segmentation program described in Sections B.3.2 and B.4

The program that converts a mask datfile into an index datfile is located in

`/mas/garden/grad/echalom/src/segment/datmask2indxmask`

and has only two parameters

`datmask2indxmask <maskin><maskout>`

The `<maskin>` parameter is an input mask datfile. Values for the input mask datfile range from 0 to R, where R is the number of regions, and 0 corresponds to samples that are not labeled. The size of the input file must be one channel, one frame, and the number of rows and columns corresponding to the size of the original image from where the training points originated. The `<maskout>` is an index datfile format. As described above, the index datfile has 3 columns (corresponding to the column, row, and region information) and P rows, corresponding to the number of labeled points in the mask datfile.

The inverse program that converts an index datfile to a mask datfile is located in

`/mas/garden/grad/echalom/src/segment/indxmask2datmask`

and has the following parameters

`indxmask2datmask <maskin><maskout>[-dims nrows ncols]`

The `<maskin>` and `<maskout>` parameters are reversed from the previous program. Namely, the `<maskin>` parameter specifies the index datfile name, and `<maskout>` specifies the mask datfile. All the properties of the index and mask datfiles remain the same as in the previous description. In this conversion we can also specify two optional parameters, `[-dims nrows ncols]`, which dictate the size of the output mask datfile. Since the size of the original input image is not known, the size (of the mask datfile) may be specified at the command line. If the dimensions are not specified, the default is to set the number of columns to the maximum column value

contained in the first (of three) column of the index datfile, and to set the number of rows to the maximum row value found in the second (of three) column of the index datfile.

B.3.2 Tracking

The program that estimates the location of the training data (from the first frame) at successive frames is located in

`/mas/garden/grad/echalom/src/tracking/feature_track`

and has the following parameters

`feature_track <tracked_mask_datfile> <input_imagefile> <mask_indexfile>`

The `<tracked_mask_datfile>` is the name of the output datfile, and is a mask datfile containing the predicted location of the training points over the length of the image sequence. The first frame contains the location of the user-supplied training data. The dimensions of this output mask datfile is the same number of rows, columns, and frames as the input image file, but the number of channels in the output file is 1, since it is a mask datfile.

The `<input_imagefile>` is the name of the input image file, and must be an RGB sequence. The `<input_mask_indexfile>` is a one frame index datfile. and its specifications are almost identical to the index datfile that is the output of the `train_image` program, except for one thing. The range of values in the third column of the index datfile must be between 1 and R (where R is the number of regions) for the tracking program input, but the range in the output to the training program is from 0 to $R-1$. Therefore, a bias of 1 needs to be applied to the third column of the index datfile before it can be used by the tracking program.

The program internally converts the RGB input image sequence to YBR format, and the correlation on the B and R channels is done at half resolution. Both the window block size and the search range for the correlation is supplied internal to the program (so a recompile is required if those need to be changed).

B.4 PDF Estimation and Segmentation

A program that computes the segmentation directly from the training data and the feature datfiles (and computes the PDF estimates of each feature internal to the program) is located in

`/mas/garden/grad/echalom/src/multimode/sup_bestfit2`

and has the following parameters

```
sup_bestfit2 <maskin (index file)><maskout (datfile)>
[-cert_dat <certainty datfile>][-cert_thresh <certainty threshold>]
[-txtfile <text_file_name>][-verbose][-rlt (relable training points)]
[-row_pos][-col_pos][-featurefiles ... (up to 10)]
```

The parameter `<maskin>` is an input parameter and contains the name of an index datfile containing the training data information. This file must be in index file format, *e.g.* 3 columns, by P (the number of training points) rows, as described in Section B.3. The parameter `<maskout>` indicates the name of the output file that contains the mask/segmented data. This file must be in mask datfile format *e.g.* it is a displayable file with the same number of rows and columns as the original image.

The parameter `[-cert_dat <certainty datfile>]` is an optional output file name. It contains floating point values between 0 and 1 that specify the certainty of the decision classification at each sample. The size of the file is thus 1 channel by 1 frame by the same number of rows and columns as the mask datfile (and the original input image). The `[-cert_thresh <certainty threshold>]` parameter if specified, applies a threshold level of the certainty to the classification process. If the certainty value at an image pixel location is above the specified threshold, then that pixel location is labeled (based upon the segmentation process) otherwise, if the certainty value is below the threshold, the sample remains unlabeled. Valid certainty values are in the interval of 0 and 1, so if the certainty threshold is set to 1 none of the points will be labeled, and conversely, if the threshold is set to 0, all the points will be labeled. If this parameter is not specified, the default is set to 0.

The [-txtfile <text_file_name>] parameter specifies the name of an ascii text file name that would store the calculate PDF values. Recall that there is one PDF estimation for each feature at each region. Thus if there are 5 regions and 9 features, there are a total of 45 density functions. For each feature, the value of the feature, f, and the likelihood (based on the PDF estimate) that a sample will take on that feature value, $p_f(f)$ are stored at 101 evenly spaced intervals. The number of intervals was determined experimentally. The text file contains two columns: the first contains the feature value, and the second contains the probability estimate of that feature value. The number of rows in the text file is therefore $R * F * 101$, where R and F correspond to the number of regions and features, respectively, and 101 refers to the number of sampling intervals of the PDF estimate. The text file contains the entire sampled PDF estimate for each feature for each region, in that order. So the first 101 text lines corresponds to the PDF estimate of the first feature in the first region, and the next 101 text lines corresponds to the second feature of the first region, and so on.

The [-verbose] option prints relevant output statistic information for each feature in each region as the PDF estimation is calculated. Specifically, it prints out the mean and variance information, as well as the best EM fit for a multimodal Gaussian fit to the data, where the number of modes varies from 1 to M, where M depends on the length of the training data. It also prints out the entropy calculation applied to the true (or all the) training data, the assumed (or training portion of the) training data, and the test portion of the training data.

The [-rlt (relable training points)] parameter, if specified, indicates that when the program computes the classification of each sample it should not assume that the region labels in the input index datfile are correct. So the training data is used only to estimate the PDF estimate, and once this estimate is established, the training data are relabeled to the most likely region, according to the PDF estimates of each region. This is useful if the data in the index datile came from a tracking program, for example, and not directly from a user-supplied training program. Since it's possible that the region label in the index datfile is not accurate when it was pro-

duced by a tracking procedure, then specifying the [-rlt] option flags the program to recompute the region for all the samples both "prelabeled" and unlabeled. The default is to not relable the training points, so if the [-rlt] parameter is not specified, the program will label the samples specified by the index datfile to the corresponding region information contained in the index datfile.

The [-row_pos] and [-col_pos] options indicate that the row position and column position information, respectively, should be considered as features in addition to any of the files specified by the [-featurefiles ... (up to 10 filenames)] parameter, if any. Obviously, at least one feature is necessary in order to perform the PDF estimation, so if neither of the row or column position flags are specified, then at least one feature file needs to be specified. The datfile I/O library allows a maximum of 10 files to be supplied. This does not mean that the number of features is limited to 10. Rather, each feature comes from each channel of the specified featurefiles. The total number of features is therefore the sum of the channels in the supplied feature files, plus one feature for each of the row and/or column position flags specified. Typically, the programs have been run with a 3-channel color feature file (*e.g.* RGB, or YIQ or LAB), a 2,4 or 5-channel texture feature file, a 2-channel motion feature file, and both row and column position.

The program requires that a "key" definition is contained in the header of each feature file. (**Note:** A "key" is part of the datfile format that allows header information about the datfiles to be indexed. For example, datfiles have a key definition called "data_sets" which contains the number of data sets in the datfile. Similarly there are key definitions for the number of channels and the dimensions of each frame. There are also optional header information tags, such as "comment" and "history." The former describes the kind of information stored in that datfile and the latter shows a history of commands that have been run to create the datfile.) The first is called "min_val" and the second is called "max_val." Both these tags are necessary in the feature file header for the sup_bestfit2 program to run. The tags are each a string of numbers and must contain the same number of values as the number of channels passed in each of the featurefiles. The information contained in the min_val

and max_val tags corresponds to the minimum and maximum possible values for the features in the feature file. For example, if the feature file is a YIQ color file, the min_val key will contain the string "0 -128 -128" and the max_val key will contain the string "255 127 127". These numbers correspond to the minimum and maximum values of the Y, I, and Q channels respectively. The keys are generally defined in a perl or shell script using a standard dat utility called **setkey**.

When the **[-txtfile]** option is specified, the order of the features is: column position (if specified), followed by row position (if specified), followed by each channel in each of the feature files, in the order the files are supplied.

B.4.1 PDF Estimation Only

Sometimes it might make sense to breakdown the probability estimation and segmentation process into two procedures. One procedure calculates the PDF estimation for each region and stores them to a file, and the second procedure calculates the segmentation based on these files containing the PDF estimation information.

The program located in

/mas/garden/grad/echalom/src/multimode/bestfit_all_emda

calculates the PDF estimate given an index datfile and a list of feature files.

```
bestfit_all_emda <maskin (index file)> [-pdfdatfile <pdf_dat_file>]
        [-pdftxtfile <text_file_name>] [-verbose]
      [-row_pos] [-col_pos] [-featurefiles ...  (up to 10)]
```

The **<maskin>** parameter is an input parameter that specifies the index datfile containing the training data information. This parameter is identical to the input index file in the **sup_bestfit2** program. The **[-pdfdatfile <pdf_dat_file>]** is an optional parameter that specifies the name of the file to store the PDF estimation. The format of the PDF file is as follows: There are R data sets (or frames) where R corresponds to the number of regions; There are F channels, corresponding to the number of features; There are 101 rows, corresponding to the resolution of the pdf

estimate; And there are 2 columns, the first column contains the range of possible feature values at 101 evenly separated intervals, and the second column contains the relative likelihood the feature can take on such a value (at each of the 101 intervals).

The `[-pdftxtfile <text_file_name>]` is an optional parameter that specifies the name of a text file to store the PDF estimation information. The order of storage is slightly different from the output of the **sup_bestfit2** program however. In this format, we store the PDF for the first region, first feature, followed by the second region, first feature, and so on. In the **sup_bestfit2** program, however, the storing priority of the feature and region are reversed, so the PDF of the second feature, first region comes immediately after the first region first feature. There is no particular reason for the different standards, except that one can run the dat utility program called **ascii2dat** directly on the text output file from the **bestfit_all_emda** program.

If neither the `[-pdfdatfile]` or `[-pdftxtfile]` parameters are specified, the PDF information is printed to standard output. The `[-verbose]`, `[-row_pos]`, `[-col_pos]` and `[-featurefiles]` optional parameters work exactly the same as explained in the **sup_bestfit2** program, except that content of what gets printed varies slightly.

B.4.2 Segmentation from PDF Files

Given a PDF datfile or set of PDF datfiles, the program

`/mas/garden/grad/echalom/src/multimode/sup_bestfit_pdf`

produces a segmentation output based upon a set of PDF datfiles and their corresponding feature files. The specific usage parameters are as follows:

```
sup_bestfit_pdf <maskout (dat file)> [-maskin <mask index file>]
    [-pdf_dat <pdf_datfile>] [-pdf_txt <pdf textfile>]
[-cert_dat <certainty datfile>] [-cert_thresh <certainty threshold>]
        [-col_pos] [-row_pos] [-infiles ...  (up to 10)]
```

The `<maskout>` parameter specifies the name of the output mask datfile. At each pixel in the image space, it contains a value between 0 and R corresponding to the

most likely region for that pixel. 0 corresponds to an unlabeled sample in cases where user requests that not all pixels need be labeled. The [-maskin] parameter is an option that allows you to provide an index datfile name, usually containg the location of training data or tracking data. Note that the index datfile is optional since the training points that were used to estimate the PDF do not necessarily come from the same frame we are trying to label.

The [-pdf_dat] option and [-pdf_txt] options are outputs containing the PDF information in datfile and text file formats, respectively. The datfile format is in the same format as the PDF datfile output of the bestfit_all_emda program. And the text file format is the same as the PDF text file output of the sup_bestfit2 program. There is almost no need whatsoever for the PDF datfile format, since it is almost identical to the PDF datfile input, except that the input may be supplied as multiple files, and the output PDF is stored as one file with F datasets and R channels (where F and R represent the number of features and regions, respectively).

The [-cert_dat] and [-cert_thresh] optional parameters are the same as explained in the sup_bestfit2 program, as are the [-row_pos] and [-col_pos]. The [-infiles] parameter is used to specify both the names of the feature files and the names of the PDF datfiles. The program internally checks if the number of columns in the input file is equal to 2, then it considers the input file to be a PDF file, other wise it considers the input file to be a feature file. The total sum of the number of data sets in all the input PDF files, must be equal to the total sum of the number of channels in all the input feature files plus the number of position features specified (*e.g.* row position and column position). Also, the number of channels (corresponding to the number of regions) in each of the input PDF files must reconcile with each other. And the dimension (number of rows and columns) of each of the input feature files must also reconcile with each other.

B.5 Post-Processing

There are two post-processing programs that take advantage of the certainty datfile information. The first program

`/mas/garden/grad/echalom/src/multimode/declassify`

is a rather simple program that unassigns all the points below a specific certainty threshold. Thus experiments could be run to measure the tradeoff between percentage of points labeled and the percentage of accurately labeled points. It has the following usage:

`declassify <maskin (mask dat file)><certdat (certainty file)><cert_thresh (new certainty threshold)><maskout (dat file)>`

The `<maskin>` parameter specifies the name of the input index datfile. To use the mask output file from the segmentation program (`sup_bestfit2` for example) it must first be converted to an index datfile format using the conversion utility `datmask2indxmask`, described at the end of Section B.3.1. The `<certdat>` parameter specifies the name of the input certainty datfile. This contains the certainty values of the labeled pixels. The `<cert_thresh>` parameter contains the new threshold value, which should obviously be higher than the threshold that was used to calculate the datfile containing the certainty measurment, otherwise there will be no change between input and output mask datfile. The `<maskout>` parameter specifies the name of the output mask datfile.

A second program called

`/mas/garden/grad/echalom/src/multimode/neighbor_fill2`

takes as input a mask datfile with contains almost all of the points labeled, but some of them unlabeled, and assigns the remaining points to the region corresponding with the most popular region amongst its neighbors. The program has the following usage:

`neighbor_fill2 <maskin (mask dat file)> <maskout (dat file)> [-pooled neighborhood size]`

The <maskin> and <maskout> parameters represent the name of the input and output mask datfile, respectively. And the [-pooled neighborhood size] option sets the starting radius of the neighborhood size. At every unlabeled point, each neighbor (within a neighborhood size) counts as one vote towards the region it belongs to, and the current unlabeled point is set to region with the most votes. If the entire neighborhood is unlabeled, the radius of the region size is incremented slowly, until a minimum of one neighbor is found. In the event of a tie, priority arbitrarily goes to the lower region number.

Bibliography

[1] Adelson, Edward, H., "Layered representations for image coding," MIT Media Laboratory Vision and Modeling Group, TR #181, December 1991.

[2] Adelson, Edward, H., and John Y. A. Wang, "Spatio-Temporal Segmentation of Video Data," MIT Media Laboratory, *Vision and Modeling Group*, TR #262, 1994.

[3] Adelson, Edward, H., and John Y. A. Wang, "Representing Moving Images with Layers," MIT Media Laboratory, *Perceptual Computing Section*, TR #279, 1994.

[4] Adelson, Edward, H., and John Y. A. Wang, "Layered Representations for Image Sequence Coding," MIT Media Laboratory Vision and Modeling Group, *Internal Report*, 1993.

[5] Adelson, Edward, H., and John Y. A. Wang, "Layered Representations for Motion Analysis," *Proceedings of Computer Vision and Pattern Recognition Conference*, June 1993.

[6] Adelson, Edward, H., Ujjaval Desai and John Y. A. Wang, "Applying Mid-level Vision Techniques for Video Data Compression and Manipulation," MIT Media Laboratory, *Vision and Modeling Group*, TR #263, 1994.

[7] Adelson, Edward, H., and William T. Freeman, "The Design and Use of Steerable Filters," *IEEE T-PAMI*, Vol. 13 #9, September 1991: 891-906.

[8] Agamanolis, S. and V. M. Bove, Jr., "Multilevel Scripting for Responsive Multimedia," *IEEE Multimedia* Vol. 4 #4, October-December 1997: 40-50.

[9] Ahuja, Narendra, Thomas S. Huang, and Juyang Weng, "Motion and Structure from Line Correspondences: Closed-Form Solution, Uniqueness, and Optimization," *IEEE T-PAMI*, Vol.14 #6, March 1992: 318-336.

[10] Ahuja, Narendra, Thomas S. Huang, and Juyang Weng, "Optimal Motion and Structure Estimation," *IEEE T-PAMI*, Vol.15 #9, September 1993: 864-884.

[11] Andrey, Phillipe and Phillipe Tarroux, "Unsupervised Image Segmentation Using A Distributed Genetic Algorithm," *Pattern Recognition*, Vol. 27 #5, May 1994: 659-673.

[12] Aviv, Gilad, "Inherent Ambiguities in Recovering 3-D Motion and Structure from a Noisy Flow Field," *IEEE T-PAMI*, Vol.11 #5, May 1989: 477-489.

[13] Becker, Shawn, Edmond Chalom and Steve Mann, "Rough Notes on Image Compositing Based on 2-D Models," MIT Media Laboratory, *Internal Report*, April 1993.

[14] Bender, Walter, and Laura Teodioso, "Salient Stills From Video," MIT Media Laboratory, *Internal Report*, May 1992.

[15] Bergen, James, R., and Michael S. Landy, "Computational Modeling of Visual Texture Segregation," from *Computational Models of Visual Processing*, by Michael S. Landy and J. Anthony Movshon. The MIT Press, 1994: 253-271.

[16] Bergen , James R., Peter J. Burt, Rajesh Hingorani and Shmuel Peleg, "A Three-Frame Algorithm for Estimating Two-Component Image Motion," *IEEE T-PAMI*, Vol.14 #9, September 1992: 886-896.

[17] Bergen , James R., Peter J. Burt, Rajesh Hingorani and Shmuel Peleg, "Multiple Component Image Motion: Motion Estimation," David Sarnoff Research Center, *Technical Report Draft*, January 1990.

[18] Bergen, J. R., and R. Hingorani, "Hierarchical Motion-Based Frame Rate Conversion," David Sarnoff Research Center, Technical Report, April 1990.

[19] Berkmann, Jens and Terry Caeli, "Computation of Surface Geometry and Segmentation Using Covariance Techniques," *IEEE T-PAMI*, Vol. 16 #11, November 1994: 1114-1116.

[20] Bieri, Hanspeter, Denise Mayor and John Wiley, "A Ternary Tree Representation of Generalized Digital Images," *Virtual World and Multimedia*, 1993: 23-35.

[21] Biggar, M.J., A. G. Constantinides and O. J. Morris, "Segmented-Image Coding: Performance Comparison with the Discrete Cosine Transform," *IEE Proceedings*, Vol. 135 #2, April 1988: 121-132.

[22] Bigun, J., and P. Schroeter, "Hierarchical Image Segmentation by Multi-Dimensional Clustering and Orientation-Adaptive Boundary Refinement," *Pattern Recognition*, Vol. 28 #5, May 1995: 695-709.

[23] Bouthemy, Patrick and Edouard Francois, "Motion Segmentation and Qualitative Dynamic Scene Analysis from an Image Sequence," *IJCV*, Vol.10 #2, 1993: 157-182.

[24] Bouthemy, Patrick and Fabrice Heitz, "Multimodal Estimation of Discontinuous Optical Flow Using Markov Random Fields," *IEEE T-PAMI*, Vol. 15 #12, December 1993: 1217-1232.

[25] Bouthemy, Patrick and Francois G. Meyer, "Region-Based Tracking Using Affine Motion Models in Long Image Sequences," *CVGIP, Image Understanding*, Vol. 60 # 2, September 1994: 119-140.

[26] Bove, V. Michael, Jr. and Edmond Chalom, "Open Architecture Television for Motion-Compensated Coding," *SPIE-VCIP*Vol. #1818, 1992: 1088-1090.

[27] Bove, V. Michael, Jr. and Edmond Chalom "Segmentation of an Image Sequence Using Multi-Dimensional Image Attributes," *Proceedings ICIP-96*, 1996: 525-528.

[28] Bove, V. Michael, Jr. and Edmond Chalom, "Segmentation of Frames in a Video Sequence Using Motion and Other Attributes," *SPIE-Digital Video Compression*, Vol. #2419, 1995: 230-241.

[29] Bozdagi, Gozde, Eli Saber, and A. Murat Tekalp, "Fusion of Color and Edge Information for Improved Segmentation and Edge Linking," University of Rochester, *Department of Electrical Engineering and Center for Electronic Imaging*, July 1995.

[30] Breeuwer, Marcel, "Motion-Adaptive Sub Band Coding of Interlaced Video," *SPIE-VCIP*, Vol. #1818, 1992: 265-275.

[31] Breipohl, A. M., and K. Sam Shanmugan, *Random Signals: Detection, Estimation and Data Analysis*, John Wiley and Sons, 1988.

[32] Buhmann, Joachim and Hansjorg Klock, "Multidimensional Scaling by Deterministic Annealing," `http://www-dbv.informatik.unibonn.de/abstracts/klock.emmcvpr.html`, March 1997.

[33] Buhmann, Joachim and Thomas Hofmann, "Pairwise Data Clustering by Deterministic Annealing," *IEEE T-PAMI*, Vol. 19 # 1, January, 1997: 1-14.

[34] Buschmann, Ralf, "Improvement of Optical Flow Estimation by HCF Control and Hierarchical Blockmatching," *IEEE Workshop on Visual Signal Processing and Communication*, 1992: 270-273.

[35] Calway, Andrew, D., Edward R. S. Pearson and Roland Wilson, "A Generalized Wavelet Transform for Fourier Analysis: The Multiresolution Fourier Transform and Its Application to Image and Audio Signal Analysis," *IEEE Transactions on Information Theory*, Vol. 38, #2, March 1992: 674-690.

[36] Chanbari, Mohammad, and Vassilis Seferidis, "Generalized Block Matching Motion Estimation," *SPIE-VCIP*, Vol. #1818, 1992: 110-119.

[37] Chellappa, Rama, David Doermann and Kamran Etemad, "Multiscale Segmentation of Unstructured Document Pages Using Soft Decision Integration," *IEEE T-PAMI*, Vol. 19 # 1, January 1997: 92-96.

[38] Chen, Yan Qui, Mark S. Nixon and David W. Thomas, "Statistical Geometrical Features for Texture Classification," *Pattern Recognition*, Vol. 28 #4, April 1995: 537-552.

[39] Chen-Chau Chu and J. K. Aggarwal, "The Integration of Image Segmentation Maps Using Region and Edge Information," *IEEE T-PAMI*, Vol. 15 #12, December, 1993: 1241-1252.

[40] Cheng, Jin-Chang and Hon-Son Don, "Segmentation of Bilevel Images Using Mathematical Morphology," *IJPRAI*, Vol.6 #4, 1992: 595-628.

[41] Choi, Jae Gark, Seong-Dae Kim and Si-Woong Lee, "Spatio-Temporal Video Segmentation Using a Joint Similarity Measure," *IEEE Trans. Circuits and Systems for Video Technology*, Vol. 7, # 2, April 1997: 279-286.

[42] Connolly, C. Ian, and J. Ross Stenstrom, "Constructing Object Models from Multiple Images," *IJCV*, Vol.9 #3, 1992: 185-212.

[43] Dempster, A. P., N. M. Laird, and D. B. Rubin, "Maximum Likelihood from Incomplete Data Via the EM Algorithm," *Journal of the Royal Statistical Society: Series B*, Vol. 39, 1977: 1-38.

[44] Deng, Yining, W. Y. Ma, and B. S. Manjunath, "Tools for Texture/Color Based Search of Images," *SPIE*, Vol. 3016, YEAR: 496-507.

[45] Drake, Alvin W., *Fundamentals Of Applied Probability Theory*, McGraw-Hill Book Company, 1967.

[46] Duda, Richard O. and Peter E. Hart, *Pattern Classification And Scene Analysis*, John Wiley and Sons, 1973.

[47] Dufaux, Frederic, Andrew Lippman and Fabrice Moscheni, "Spatio-Temporal Segmentation Based on Motion and Static Segmentation. MIT Media Laboratory, *Internal report*, 1995.

[48] Dufaux, Frederic and Murat Kunt, "Multigrid Block Matching Motion Estimation with an Adaptive Local Mesh Refinement," *SPIE-VCIP*, Vol. # 1818, 1992: 97-109.

[49] Duncan, James, S., Lawrence H. Staib, "Boundary Finding with Parametrically Deformable Models," *IEEE T-PAMI*, , Vol.14 #11, November 1992: 1061-1075.

[50] Ebrahimi, Touradj and Murat Kunt, "Image Sequence Coding Using a Three Dimensional Wavelet Packet and Adaptive Selection," *SPIE-VCIP*, Vol. # 1818, 1992: 222-232.

[51] Etoh, Minoru and Yoshiaki Shirai, "Segmentation and 2D Motion Estimation by Region Fragments," *ICCV*, February 1993: 192-199.

[52] Faugeras, Olivier, D. and Zhengyou Zhang, "Three-Dimensional Motion Computation and Object Segmentation in a Long Sequence of Stereo Frames," *IJCV*, Vol.7 #3, 1992: 212-241.

[53] Forchheimer, R., Haibo Li, and A. Lundmark, "Motion Compensated Alignment and Its Application to Interframe Image Coding," *Image Coding Group*, Department of Electrical Engineering, Linkoping University, Sweden, 1993

[54] Fox, Geoffrey, Eitan Gurewitz and Kenneth Rose, "Constrained Clustering as an Optimized Method," *IEEE T-PAMI*, Vol. 15 # 8, August, 1993: 785-794.

[55] Freeman, K., R. Jackson, and L. MacDonald, *Computer Generated Color: A Practical Guide to Presentation and Display*, John Wiley and Sons, 1994.

[56] Fuh, Chiou-Shann and Petros Maragos, "Affine Models for Motion and Shape Recovery," *SPIE-VCIP*, Vol. #1818, 1992: 120-134.

[57] Geiger, Davi and Alan Yuille, "A Common Framework for Image Segmentation," *IJCV*, Vol. 6 #3, 1991: 227-243.

[58] Gersho, A., D. Miller, A. Rao, and K. Rose, "A Global Optimization Technique for Statistical Classifier Design," *IEEE Transactions on Signal Processing*,, Vol.44, # 12, December 1996: 3108-3122.

[59] Gersho, A., D. Miller, A. Rao, and K. Rose, "Deterministically Annealed Mixture of Experts Models for Statistical Regression," *ICASSP*, 1997.

[60] Giroso, Frederico and Jose L. Marroquin, "Some Extensions of the K-Means Algorithm for Image Segmentation and Pattern Classification," MIT Artificial Intelligence Laboratory, *AI Memo #1390*, January 1993.

[61] Gray, Robert, M., and Karen L. Oehler, "Combining Image Compression and Classification Using Vector Quantization," *IEEE T-PAMI*, Vol. 17 #5, May 1995: 461-473.

[62] Harashima, Hiroshi, Nakaya, Yuichiro "An Iterative Motion Estimation Method Using Triangular Patches for Motion Compensation," *SPIE-VCIP*, Vol. #1605, 1991: 546-557.

[63] Horn, B.K.P., *Robot Vision*, MIT Press and McGraw Hill, 1986.

[64] Horn, B.K.P., and B.G. Schunck, "Determining Optical Flow," MIT Artificial Intelligence Laboratory, *AI Memo # 572*, April 1980.

[65] PLACEBO

[66] Jain, Anil, K., and Jianchang Mao, "Texture Classification and Segmentation Using Multiresolution Simultaneous Autoregressive Models," *Pattern Recognition*, Vol.25 #2, February 1992: 173-188.

[67] Jolion, J.M., and A. Montanvert, "The Adaptive Pyramid: A Framework for 2D Image Analysis," *CVGIP, Image Understanding*, Vol.55 #3, May 1992: 339-348.

[68] Kan, Sing Bing, Richard Szeliski, "Recovering 3D Shape and Motion from Image Streams Using Non-Linear Least Squares," DEC, Cambridge Research Laboratory, *Technical Report*, March 1993.

[69] Kanade, Takeo and Bruce D. Lucas, "An Iterative Image Registration Technique with an Application to Stereo Vision," *Proceedings DARPA Image Understanding Workshop*, 1981: 121-130.

[70] Kermode, Roger G., "Image Representation for Vision, Background Extraction," MIT Media Laboratory, *Final Project for 4.906*, December 1992.

[71] Kermode, Roger G, "Requirements for Real Time Digital Video Compression," Report for *Silicon Graphics Inc.*, July 1993.

[72] Kermode, Roger and Andrew Lippman, "Generalized Predictive Coding of Movies," MIT Media Laboratory, *Internal Report*, December 1992.

[73] Kingsbury, Nick, G. and Robert W. Young, "Video Compression Using Lapped Transforms for Motion Estimation/Compensation and Coding," *SPIE-VCIP*, Vol. #1818, 1992: 276-288.

[74] Labit, C. and H. Nicolas, "Region-Based Motion Estimation Using Deterministic Relaxation Schemes for Image Sequence Coding," *ICASSP*, 1992: 265-268.

[75] Lee, Lillian, Fernando Pereira and Naftali Tishby, "Distributional Clustering of English Words," `http://xxx.lanl.gov/abs/cmp-lg/9408011`, August 1994.

[76] Lee, Shinhak and Shahriar Negahdaripour, "Motion Recovery from Image Sequences Using Only First Order Optical Flow Information," *IJCV*, Vol.9 #3, 1992: 163-184.

[77] Lippman, Andrew, "Coding of Motion Pictures and Associated Audio," MIT Media Laboratory, *ISO/IEC et al.*, February 1993.

[78] Liu, Bede, Boon-Lock Yeo and Minerva Yeung, "Extracting Story Units from Long Programs for Video Browsing and Navigation," *International Conference on Multimedia Computing and Systems*, June 1996.

[79] Liu, F.and R. W. Picard, "Periodicity, Directionality, and Randomness: Wold Features for Image Modeling and Retrieval," MIT Media Laboratory, *Perceptual Computing Section*, TR #320, 1995.

[80] Liu, Fang and Rosalind W. Picard, "A New WOLD Ordering for Image Similarity," MIT Media Laboratory, *Perceptual Computing Section*, TR #237, April 1994.

[81] Minka, T. P., and Rosalind W. Picard, "Vision Texture for Annotation," MIT Media Laboratory, *Perceptual Computing Section*, TR #302, 1994.

[82] Mohan, Rakesh, and Ramakant Nevatia, "Perceptual Organization for Scene Segmentation and Description," *IEEE T-PAMI*, Vol.14 #6, June 1992: 616-635.

[83] Neal, Radford M. and Geoffrey E. Hinton, "A View of the EM Algorithm That Justifies Incremental, Sparse, and Other Variants," University of Toronto Computer Science Department, 1998. To appear in *Learning in Graphical Models*, Kluwer Academic Press.

[84] Orchard, Michael, T., and Gary J. Sullivan, "Overlapped Blocked-Motion Compensation: An Estimation-Theoretic Approach," PictureTel Co., Technical Report submitted to *IEEE T-IP*, November 1993.

[85] Pentland, Alex, P., and Stan Sclaroff, "Object Recognition and Categorization Using Modal Matching," MIT Media Laboratory, *Vision and Modeling Group*, TR #267, 1994.

[86] Pentland, Alex, P., and Stan Sclaroff, "Modal Matching for Correspondence and Recognition," MIT Media Laboratory, *Perceptual Computing Section*, TR #201, 1993.

[87] Pentland, Alex, P., and Stan Sclaroff, "Photobook: Tools for Content-Bases Manipulation of Image Databases," MIT Media Laboratory, *Perceptual Computing*, TR #255, 1994.

[88] Picard, Rosalind W., "Random Field Texture Coding," MIT Media Laboratory, *Vision and Modeling Group*, TR #185, 1992.

[89] Picard, Rosalind W., "Structured Patterns from Random Fields," MIT Media Laboratory, *Perceptual Computing Group*, TR #200, 1992.

[90] Picard, Rosalind, W., and Kris Popat, "Exaggerated Consensus in Lossless Image Compression," MIT Media Laboratory, *Perceptual Computing Section*, TR #289, November, 1994.

[91] Poggio, Tomaso, and Sebastian Toelg, "Towards an Example-Based Image Compression Architecture for Video-Conferencing. MIT Artificial Intelligence Laboratory, *AI Memo #1494*, June 1994.

[92] Redner, Richard and Homer E. Walker, "Mixture Densities, Maximum Likelihood and the EM Algorithm," *SIAM Review*, Vol. 26 #2, April 1984: 195-239.

[93] Rissanen, Jorma, "Minimum Description Length Principle," *Encyclopedia of Statistical Sciences*, Vol. 5, 1982: 523-527.

[94] PLACEBO

[95] Therrien, Charles W., *Decision Estimation And Classification*, John Wiley and Sons, 1989.

[96] Veijanen, Ari, "Unsupervised Image Segmentation Using an Unlabeled Region Process," *Pattern Recognition*, Vol. 27 #6, June 1995: 841-852.

[97] Wegmann, B. and C. Zetzsche, "Efficient Image Sequence Coding by Vector Quantization of Spatiotemporal Bandpass Outputs," *SPIE-VCIP*, Vol. #1818, 1992: 1146-1154.

[98] Yuille, Alan and Song Chun Zhu, "Region Competition: Unifying Snakes, Region Growing, and Bayes/MDL for Multiband Image Segmentation," *IEEE T-PAMI*, Vol. 18, #9, September 1996: 884-900.

[99] Vasconcelos, Nuno, "Digital Image Representations," MIT Media Laboratory, *Internal report*, August 1995.

[100] Weiss, Yair and E.H. Adelson, "Perceptually Organized EM: A Framework for Motion Segmentation That Combines Information About Form and Motion," MIT Media Laboratory, *Perceptual Computing Section*, TR #315, 1995.

THESIS PROCESSING SLIP

FIXED FIELD: ill. _____ name _____

index _____ biblio _____

▶ COPIES: (Archives) Aero Dewey (Eng) Hum

Lindgren Music Rotch Science

TITLE VARIES: ▶☐ _____

_____

_____

NAME VARIES: ▶☐ _____

_____

IMPRINT: (COPYRIGHT) _____

▶ COLLATION: 202 l

_____

▶ ADD. DEGREE: _____ ▶ DEPT.: _____

SUPERVISORS: _____

_____

NOTES:

cat'r: date:

page: F48

▶ DEPT: E.E.

▶ YEAR: 1998 ▶ DEGREE: Ph.D.

▶ NAME: CHALOM, Edmond